TABFM-AUTO: SELF-EVOLVING PIPELINES FOR TABULAR FOUNDATION MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Tabular foundation models achieve strong zero-shot accuracy on structured data by pretraining on synthetic tables, but they ignore the column names, task descriptions, and auxiliary files that carry dataset semantics. Meanwhile, self-evolving machine learning engineering (MLE) agents train models from scratch on each dataset, yet jointly searching over features, architectures, and hyperparameters is noisy and prone to overfitting. We introduce `TabFM-Auto`, which pairs a tabular foundation model, `TabFM`, with a language model agent that evolves the data pipeline around it. Guided by dataset metadata and validation feedback, `TabFM-Auto` iteratively refines data cleaning, feature engineering, context selection, and post-processing to reduce `TabFM`'s error. Across all 51 datasets of the TabArena benchmark, five `TabFM-Auto` configurations with different agents and language models take the top five overall positions, and the best raises `TabFM` from 1785 to 2013 Elo. The discovered pipelines also transfer to other frozen tabular foundation models (+69 to +143 Elo) with no further search. On the 8 tabular competitions of MLE-Bench, `TabFM-Auto` ranks first overall among MLE agents.

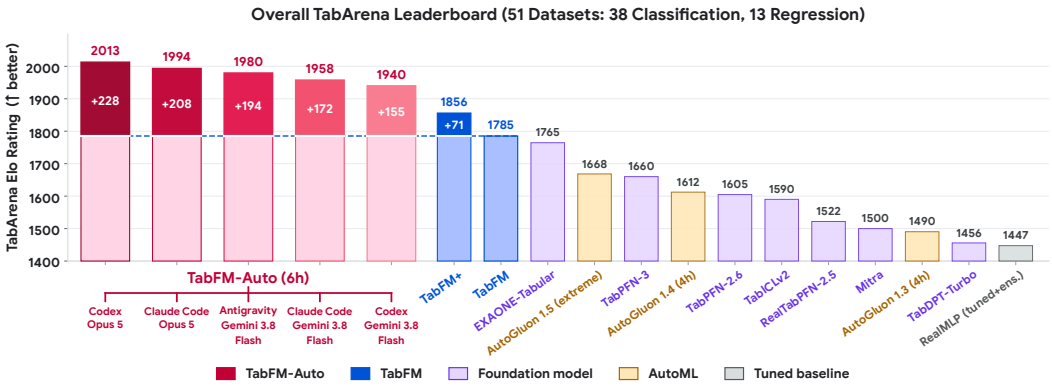

Figure 1: **Overall TabArena leaderboard across all 51 datasets (38 classification, 13 regression).** All five `TabFM-Auto` configurations (red) take the top five positions, reaching up to 2013 Elo (+228 over the `TabFM` baseline) and outperforming all other tabular foundation models.

1 INTRODUCTION

Tabular Foundation Models (TFMs) such as TabPFN (Hollmann et al., 2022; 2025), TabICL (Qu et al., 2025; 2026), and TabFM (Kong et al., 2026) pretrain a transformer on synthetic datasets so it can predict on new tables in a single forward pass without gradient updates (Müller et al., 2022). Although TFMs match or outperform tuned gradient-boosted decision trees (GBDTs) (Chen & Guestrin, 2016; Ke et al., 2017; Prokhorenkova et al., 2018), their attention layers see only floating-point magnitudes and category indices. For example, they treat a column named `systolic_bp` or `ICD9_diagnosis` the same as an anonymous column `X_17`. Thus, a TFM cannot infer domain formulas from column names, group clinical codes by hierarchy, or recognize when a number such as `0` encodes a missing measurement. Prior work (`TabFM+`, Kong et al., 2026) extends `TabFM` with

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

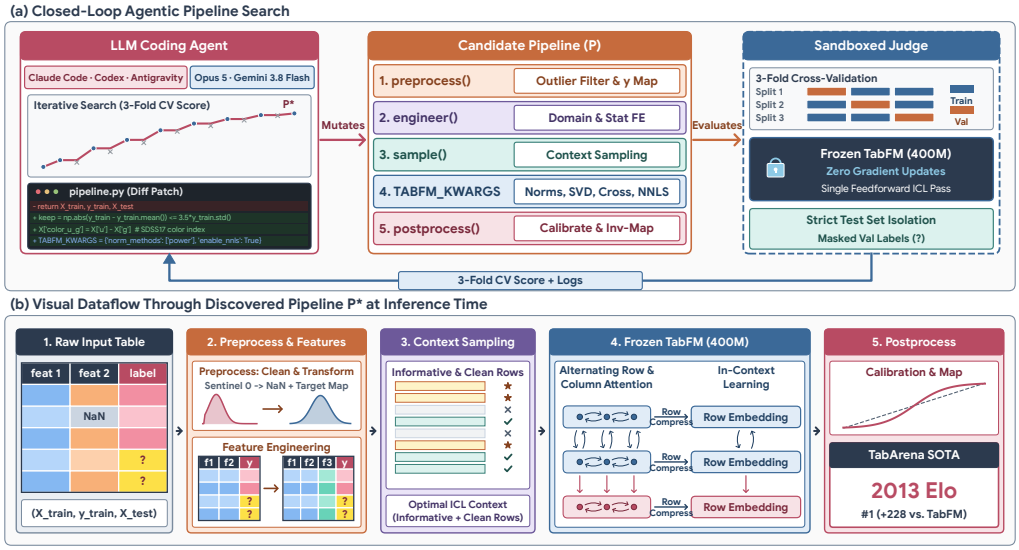

Figure 2: **Overview of `TabFM-Auto`.** (a) Within a sandboxed evaluator, a coding agent repeatedly edits the data pipeline P around a frozen 400M-parameter `TabFM` model using 3-fold cross-validation on the training set. (b) At test time, the selected pipeline P^* transforms the input table through data cleaning, feature engineering, context sampling, frozen `TabFM` inference, and post-processing.

pairwise cross features, singular value decomposition (SVD) projections, and multi-view ensembling, but these generic operations ignore what the columns represent.

Pretrained on scientific text and code, Large Language Models (LLMs) understand the domain meaning of column names and task descriptions, yet they perform poorly as direct tabular predictors due to context-length limits, coarse number tokenization (Hegselmann et al., 2023; Zhou et al., 2024; 2026; Fu et al., 2026), and miscalibrated probabilities. Existing machine learning engineering (MLE) agents (Jiang et al., 2025; Yang et al., 2025; Du et al., 2026) instead use LLMs to edit end-to-end training scripts that fit tree ensembles or neural networks from scratch during search. However, searching over features, model architectures, and hyperparameters at the same time is noisy: training variance often masks small feature gains and leads to validation overfitting.

We introduce `TabFM-Auto`, which pairs an LLM agent with a frozen tabular foundation model, `TabFM`. Guided by dataset metadata, the agent iteratively edits a data pipeline for data cleaning, feature engineering, context selection, and post-processing via 3-fold cross-validation on the training set while keeping `TabFM` fixed. Since `TabFM` requires no gradient training, each candidate pipeline is fast to evaluate and free of model-retraining noise. After search finishes, we evaluate the best pipeline once on the official test set. The pipeline also selects context rows on large or imbalanced tables and calibrates probabilities to skewed class priors.

We test five `TabFM-Auto` configurations that combine three agent harnesses, `Claude Code` (`CC`), `Antigravity` (`AGY`), and `Codex`, with two language models (`Claude Opus 5` and `Gemini 3.8 Flash`). On all 51 datasets of the TabArena benchmark (Erickson et al., 2026), they take the top five overall positions and outperform tuned AutoGluon ensembles and all evaluated tabular foundation models. The top configuration (`Codex` with `Opus 5`) raises `TabFM` from 1785.3 to 2013.0 Elo. The discovered pipelines transfer without further search to other tabular foundation models (up to a +143.3 Elo improvement). On the 8 tabular competitions of MLE-Bench (Chan et al., 2025), `TabFM-Auto` ranks first overall ahead of existing MLE agents.

2 Related Work

Tabular Foundation Models and In-Context Optimization. Prior-Data Fitted Networks (Müller et al., 2022) pretrain transformers on synthetic datasets so that a single forward pass approximates

Table 1: **Side-by-side TabArena Classification (38 datasets) and Regression (13 datasets) leaderboards.** Each `TabFM-Auto` configuration is evaluated individually against the 66-method TabArena pool (Erickson et al., 2026): **Elo** measures pairwise rating, **Wins** counts datasets where a method ranks first, **Improv.** measures mean relative error gap to the per-dataset best model, and **G-Mean** is the geometric mean test error across datasets.

(a) Classification (38 Datasets)

#	Method	Elo ↑	Wins ↑	Improv. ↓	G-Mean ↓
1	TabFM-Auto (Codex, Opus 5)	**1966.3**	13.84	1.91%	0.0913
2	TabFM-Auto (AGY, Gemini 3.8 Flash)	1937.5	**16.11**	2.45%	**0.0902**
3	TabFM-Auto (CC, Opus 5)	1933.9	14.93	**1.69%**	0.0913
4	TabFM-Auto (CC, Gemini 3.8 Flash)	1918.7	13.26	2.52%	0.0928
5	TabFM-Auto (Codex, Gemini 3.8 Flash)	1875.0	15.44	2.43%	0.0924
6	TabFM+	1836.7	12.22	2.91%	0.0948
7	TabFM	1768.7	7.91	3.80%	0.0957
8	EXAONE-Tabular	1759.5	4.39	7.28%	0.1009
9	AutoGluon 1.5 (extreme)	1664.6	1.72	7.86%	0.1010
10	TabPFN-3	1637.7	0.87	10.18%	0.1050
11	AutoGluon 1.4 (4h)	1617.1	0.51	11.24%	0.1079
12	TabPFN-2.6	1586.9	0.14	11.72%	0.1075
13	TabICLv2	1583.6	1.08	10.88%	0.1060
14	RealTabPFN-2.5	1533.5	0.07	12.11%	0.1077
15	AutoGluon 1.3 (4h)	1473.4	0.14	14.24%	0.1147
16	RealMLP (tuned+ens.)	1435.2	0.19	15.37%	0.1161

(b) Regression (13 Datasets)

#	Method	Elo ↑	Wins ↑	Improv. ↓	G-Mean ↓
1	TabFM-Auto (Codex, Opus 5)	**2512.9**	**11.12**	**0.00%**	**15.86**
2	TabFM-Auto (CC, Opus 5)	2489.7	10.51	0.00%	15.89
3	TabFM-Auto (Codex, Gemini 3.8 Flash)	2423.2	9.58	0.01%	15.93
4	TabFM-Auto (CC, Gemini 3.8 Flash)	2407.9	9.41	0.01%	15.95
5	TabFM-Auto (AGY, Gemini 3.8 Flash)	2347.8	9.98	0.00%	15.88
6	TabFM+	2169.2	5.33	0.14%	16.17
7	TabFM	2045.9	1.80	1.43%	16.40
8	EXAONE-Tabular	1967.8	1.17	2.46%	16.58
9	TabPFN-3	1863.1	0.71	2.15%	16.51
10	AutoGluon 1.5 (extreme)	1851.2	0.39	3.67%	16.78
11	TabPFN-2.6	1790.1	0.13	3.74%	16.79
12	AutoGluon 1.4 (4h)	1731.0	0.07	4.42%	16.91
13	TabICLv2	1726.2	0.81	3.68%	16.78
14	TabDPT-Turbo	1661.9	0.26	4.93%	17.02
15	AutoGluon 1.3 (4h)	1646.1	0.03	6.01%	17.23
16	RealMLP (tuned+ens.)	1622.8	0.07	5.41%	17.09

Bayesian in-context prediction (Garg et al., 2022; Von Oswald et al., 2023; Li et al., 2023; Fu et al., 2024). TabPFN (Hollmann et al., 2022) introduced this paradigm for small classification tables. Later work extended it to regression and larger contexts (Hollmann et al., 2025; Grinsztajn et al., 2025; 2026), added continued pretraining on real data (Ma et al., 2025; Garg et al., 2025; Hosseinzadeh et al., 2026), and made attention linear-time with inducing points (Qu et al., 2025; 2026). Since these models are pretrained on anonymous synthetic variables, they typically ignore column names and task descriptions at test time. Even `TabFM+` (Kong et al., 2026) only adds generic feature crosses, SVD projections, and view ensembling. Other work adds language encoders to tabular models to read column names or text cells (Kim et al., 2024; Yan et al., 2024; Tajjar et al., 2026). Rather than modifying the tabular foundation model architecture, `TabFM-Auto` keeps `TabFM` frozen and uses an LLM agent to turn column names and task metadata into explicit pipeline code.

AutoML, LLM Feature Engineering, and MLE Agents. Supervised learning on tabular data has traditionally relied on GBDTs (Chen & Guestrin, 2016; Ke et al., 2017; Prokhorenkova et al., 2018) and deep tabular networks (Gorishniy et al., 2021; 2025). Classical automated machine learning (AutoML) systems such as Auto-sklearn (Feurer et al., 2015) and AutoGluon-Tabular (Erickson et al., 2020) automate model selection and multi-layer stacking across these estimators. Prior LLM feature-engineering methods such as CAAFE (Hollmann et al., 2023), FeatLLM (Han et al., 2024), and OCTree (Nam et al., 2024) prompt an LLM to append derived columns to single tables. `TabFM-Auto` extends this loop to the full data pipeline around an in-context model, including data cleaning, context-row selection, output calibration, and reading auxiliary files within a code-execution sandbox. Other systems automate end-to-end data science and multi-agent workflows (Huang et al., 2024; Tornede et al., 2024; Qi et al., 2026) or evolve programs through LLM-guided search (Romera-Paredes et al., 2024; Novikov et al., 2025; Xu et al., 2026), as `TabFM-Auto` does for data pipelines. End-to-end agents such as AIDE (Jiang et al., 2025), R&D-Agent (Yang et al., 2025), and MLEvolve (Du et al., 2026) retrain tree ensembles or neural networks during search, whereas `TabFM-Auto` keeps `TabFM` frozen and searches only over the data pipeline.

3 Methodology

3.1 Problem Formulation

Consider a tabular prediction task with a labeled training dataset $\mathcal{D}_{\text{train}} = (\mathbf{X}_{\text{train}}, \mathbf{y}_{\text{train}})$ of T_{train} rows and H columns, an unlabeled test dataset $\mathbf{X}_{\text{test}}$, and dataset metadata $\mathcal{M}$ containing column names, units, task descriptions, and auxiliary file schemas. A zero-shot tabular foundation model f_{θ^*} such as `TabFM` (Kong et al., 2026) takes $(\mathbf{X}_{\text{train}}, \mathbf{y}_{\text{train}})$ as in-context examples and predicts test labels $\hat{\mathbf{y}} = f_{\theta^*}(\mathbf{X}_{\text{test}} \mid \mathbf{X}_{\text{train}}, \mathbf{y}_{\text{train}})$ in a single forward pass with frozen weights θ^*.

Optimization Objective. We partition $\mathcal{D}_{\text{train}}$ into 3 internal cross-validation folds without touching the held-out test split (Figure 2). On each internal fold $(\mathcal{D}_{\text{subtrain}}, \mathcal{D}_{\text{val}})$ and conditioned on $\mathcal{M}$, the

coding agent searches over executable pipelines $P = (\Phi_{\text{clean}}, \Phi_{\text{feat}}, \mathcal{S}_{\text{ctx}}, \Psi_{\text{post}}) \in \mathcal{P}$, defined below, to maximize the mean 3-fold cross-validation metric $\mathcal{U}_{\text{val}}$ around the fixed weights θ^*:

$$P^* = \arg\max_{P \in \mathcal{P}} \mathcal{U}_{\text{val}}\Big(\Psi_{\text{post}}\big(f_{\theta^*}\big(\tilde{\mathbf{X}}_{\text{val}} \mid \mathcal{S}_{\text{ctx}}(\tilde{\mathbf{X}}_{\text{subtrain}}, \tilde{\mathbf{y}}_{\text{subtrain}})\big),\ \mathbf{y}_{\text{subtrain}},\ \tilde{\mathbf{X}}_{\text{val}}\big),\ \mathbf{y}_{\text{val}}\Big), \tag{1}$$

$$\text{where} \quad (\tilde{\mathbf{X}}_{\text{subtrain}}, \tilde{\mathbf{y}}_{\text{subtrain}}, \tilde{\mathbf{X}}_{\text{val}}) = \Phi_{\text{feat}}\big(\Phi_{\text{clean}}(\mathbf{X}_{\text{subtrain}}, \mathbf{y}_{\text{subtrain}}, \mathbf{X}_{\text{val}})\big).$$

Throughout search, θ^* stays fixed, and passing unlabeled inputs $\mathbf{X}_{\text{val}}$ (or $\mathbf{X}_{\text{test}}$) with the training split while withholding evaluation labels lets the agent perform both *inductive* and *transductive* learning.

3.2 Building Pipelines Around Frozen TabFM

Each candidate pipeline P implements the four stages in Equation (1) through four modular Python functions (`preprocess()`, `engineer()`, `sample()`, and `postprocess()`), together with the `TABFM_KWARGS` configuration dictionary passed to f_{θ^*} (Figure 2). Rather than rewriting an end-to-end training script, the agent edits only these components, and each stage addresses a specific limitation of a frozen TFM. The first two stages shape the table that `TabFM` reads. **Data cleaning and target conditioning** (Φ_{clean}) handles values whose meaning a TFM cannot read from magnitudes alone, such as a `0` that encodes a missing measurement. Using $\mathcal{M}$, `preprocess()` can recode such placeholders into missingness indicators, filter corrupted training rows, and apply reversible target transformations $g(\mathbf{y}_{\text{train}})$, such as $\log(1+y)$, that compress skewed targets. **Semantic feature engineering** (Φ_{feat}) is where the LLM's domain knowledge enters the pipeline. Given the cleaned tables, `engineer()` translates $\mathcal{M}$ into explicit columns for both inductive features (domain ratios and formulas, group statistics, and auxiliary-file summaries) and label-free transductive features over combined train and test inputs (e.g., entity-graph degrees in Figure 3). Writing a formula as one column spares `TabFM` from inferring it in context, which helps most on real-world schemas.

The remaining two stages control which rows `TabFM` conditions on and how its outputs are used. **Context selection** ($\mathcal{S}_{\text{ctx}}$) is needed because `TabFM` is pretrained on tables of up to 16,384 rows (Kong et al., 2026) and could potentially degrade in performance on much larger tables, yet uniform subsampling can drop rare classes. `sample()` therefore builds multiple context subsets (views) that stratify by class or oversample minority classes, and combining predictions across views lets `TabFM` use more rows than one context window holds. **Post-processing and output calibration** (Ψ_{post}) applies g^{-1} to return regression predictions to the original target scale, which keeps candidates with different target transformations comparable. For classification, it can shift log-odds toward the training class prior or apply temperature or Platt scaling (Platt, 1999), since oversampled views and synthetic pretraining data may not match the class balance and confidence of real data.

Generic versions of all four stages already exist in `TabFM+` (Kong et al., 2026), which extends `TabFM` to normalize inputs and clip outliers (Φ_{clean}), append pairwise feature crosses and SVD projections (Φ_{feat}), ensemble predictions over multiple views with a configurable context size ($\mathcal{S}_{\text{ctx}}$), and weight the views by non-negative least squares (NNLS) (Lawson & Hanson, 1995) before calibrating the output (Ψ_{post}). **`TABFM_KWARGS`** exposes these overlapping operations as constructor settings of f_{θ^*}. The agent can thus cover the generic parts of a stage by tuning settings instead of writing code, and use the four functions for what the `TabFM+` presets cannot express, such as domain formulas or class-balanced contexts. Keeping these settings apart from the four functions also lets us transfer the agent's searched pipelines unchanged around other TFMs (§4.1).

The agent self-evolves P by **iterative search**. Every run starts from the identity pipeline P_0 (Appendix Figure 8), which feeds the unmodified table to `TabFM` with default settings, and the agent keeps evolving P to improve the cross-validation score $\mathcal{U}_{\text{val}}$. Because θ^* stays frozen, candidates need no gradient training, no retraining noise masks small feature gains, and every improvement over P_0 comes from the agent's new pipeline. To hide test labels from agent-written code, the evaluation harness runs every candidate in a **sandbox** without network access or the held-out test splits (Figure 2a and Appendix A). After search, the harness freezes P^* and scores the official test splits.

4 Experiments

We evaluate `TabFM-Auto` on two benchmarks: against tabular foundation models and tuned AutoML ensembles on the 51-dataset TabArena benchmark (§4.1), and against MLE agents that train models from scratch on MLE-Bench-Tabular (§4.2).

I: Domain-Knowledge Features · `airfoil_self_noise` · **RMSE: 1.073 → 0.888**

```
def engineer(X_tr, y_tr, X_te):
  for df in (X_tr, X_te):
    f, u = df["frequency"], df["free-stream-velocity"]
    c, d = df["chord-length"], df["suction-side-
        thickness"]
    a = df["attack-angle"]
    df["log_st_d"] = np.log(f * d / u) # Strouhal St = f
        *d/U
    df["st_h"] = f * c * np.sin(np.radians(a)) / u
    df["helmholtz"], df["d_over_c"] = f * c / 343.0, d /
        c
  return X_tr, X_te
```

Both `Opus 5` and `Gemini 3.8 Flash` compute aeroacoustic scaling laws from the column headers.

Boundary-Layer Strouhal. Builds $St = f\delta^*/U_\infty$ (log-scale) for trailing-edge boundary-layer scaling.

Acoustic Compactness. Adds frontal Strouhal ($fc\sin\alpha/U_\infty$), Helmholtz ($He = fc/a_0$), and thickness ratio (δ^*/c).

II: Statistical & Structural Features · `Amazon_employee_access` · **1−AUROC: 0.1416 → 0.1063**

```
def engineer(X_tr, y_tr, X_te):
  cols = [c for c in X_tr.columns if c != "row_id"]
  all_df = pd.concat([X_tr[cols], X_te[cols]], axis=0)
  for df in (X_tr, X_te):
    for c in cols: # 1. Marginal entity counts (8
        features)
      df[f"{c}_cnt"] = df[c].map(all_df[c].value_counts()
          )
    for c1, c2 in PAIRS: # 2. Pairwise co-occurrences
        (13 pairs)
      vc = all_df.groupby([c1, c2]).size()
      df[f"{c1}_{c2}_cnt"] = df.set_index([c1, c2]).index
          .map(vc)
    for s, t, nm in GROUP_MAPS: # 3. Bipartite degrees
        (32 maps)
      df[f"{s}_{nm}"] = df[s].map(all_df.groupby(s)[t].
          nunique())
  return X_tr, X_te
```

Prunes duplicate `ROLE_CODE` in `preprocess()` and builds 53 label-free graph features in `engineer()`.

Marginal Frequencies. 8 entity counts across the combined train and test tables.

Pairwise Co-occurrences. 13 joint counts across ID pairs such as (role, resource).

Bipartite Degrees. 32 distinct cardinalities (`nunique`), e.g., resources per manager.

III: Context & Calibration Only · `website_phishing` · **Log-Loss: 0.2104 → 0.2024**

```
def sample(X_tr, y_tr, X_te, max_rows):
  idx = np.arange(len(X_tr))
  return [idx, np.r_[idx, np.flatnonzero(y_tr == "
      Suspicious")]]
def postprocess(pred, y_tr, X_te):
  p, cols = pred.to_numpy(), pred.columns
  pi = y_tr.value_counts(normalize=True).reindex(cols).
      values
  z = np.log(p) + np.log(pi / p.mean(0))
  q = np.exp(z - z.max(1, keepdims=True))
  return pd.DataFrame(q / q.sum(1, keepdims=True),
      columns=cols)
```

Leaves `preprocess()` and `engineer()` as pass-throughs.

Context Ensemble (`sample`). Pairs the natural draw with a 2× oversampled `Suspicious` view.

Prior Correction (`postprocess`). Shifts mean predicted probabilities ($\bar{p}$) toward training priors (π) in log-odds space.

Figure 3: **Representative `TabFM-Auto` pipelines across the three categories on TabArena (`Antigravity` with `Gemini 3.8 Flash`).** (I) Explicit aeroacoustic scaling laws on `airfoil_self_noise`, (II) relational graph features on `Amazon_employee_access`, and (III) minority-class context ensembling and prior calibration on `website_phishing`.

4.1 TabArena

Experimental Setup. We evaluate `TabFM-Auto` on all 51 datasets of the TabArena benchmark (Erickson et al., 2026) (38 classification and 13 regression), where each dataset uses 3-fold outer cross-validation repeated 3 or 10 times, yielding 9 or 30 official train/test splits (evaluation folds in Table 9). Running a separate LLM agent search on every fold would be prohibitively costly in tokens and GPU time (Table 3). Thus, we run pipeline search once per dataset using 3-fold cross-validation on the training set of fold 0 (up to 96 evaluations or 6 hours on one H100 GPU), allowing both inductive and transductive features while withholding evaluation labels. We then freeze P^*, evaluate it on the held-out test splits of all folds $k \in \{0, \ldots, S-1\}$, and report both all-folds and fold-0-only test leaderboards (Tables 4–5). We test five configurations combining three agent harnesses (`Claude Code`, `Codex`, and `Antigravity`) with two language models (`Claude Opus 5` and `Gemini 3.8 Flash`). We compare against `TabFM`, `TabFM+` (Kong et al., 2026), other tabular foundation models (`EXAONE-Tabular`, `TabPFN-3`, `TabICLv2`), and

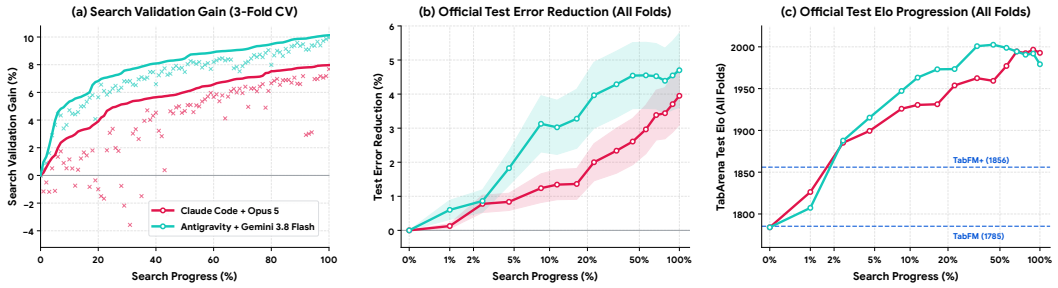

Figure 4: **Search dynamics and test generalization.** (a) 3-fold cross-validation gain on fold 0's training set, (b) official test error reduction across all folds, and (c) official test TabArena Elo progression across all folds.

4-hour `AutoGluon` ensembles (Erickson et al., 2020). Following Erickson et al. (2026), we rate each configuration individually against the 66-method TabArena pool using Bradley–Terry **Elo** (Bradley & Terry, 1952; Elo, 1978; Hunter, 2004) (anchored to default Random Forest $= 1000$), dataset **Wins**, oracle **Improvability** ($1 - \mathrm{err}_{\mathrm{best}}/\mathrm{err}_{\mathrm{method}}$), and Geometric Mean Test Error (**G-Mean**).

Main Results on TabArena. Across all 51 datasets (Figure 1, Appendix B.1, and Appendix E Table 9), our five `TabFM-Auto` configurations take the top five overall positions, reaching 2013.0 Elo (`Codex` with `Opus 5`) and outperforming 4-hour `AutoGluon 1.5 (extreme)` (1668.4 Elo) by $+272$ to $+345$ Elo. The gains hold on both task types (Table 1 and Appendix Figure 9). On the 38 classification datasets, `TabFM-Auto` raises `TabFM` by $+197.6$ Elo to 1966.3 Elo, and `Antigravity` with `Gemini 3.8 Flash` has the lowest G-Mean test error (0.0902) and the most dataset wins (16.11). On the 13 regression datasets, `TabFM-Auto` raises `TabFM` by $+467.0$ Elo to 2512.9 Elo, improving over `TabFM` on all 13 datasets and reducing oracle improvability to 0.00%. We hypothesize that regression gains are larger because linearizing physical ratios in Φ_{feat} and skewed targets in Φ_{clean} lets `TabFM` smoothly interpolate continuous functions that tree splits approximate only coarsely.

Pairwise fold win rates on the official test sets (Figure 10) agree with the Elo ranking. `Claude Code` with `Opus 5` and `Antigravity` with `Gemini 3.8 Flash` tie head-to-head (50.7% vs. 49.3%). They win against `TabFM` on 74.3% and 72.1% of test folds (91.9% and 83.3% on regression) and against `AutoGluon 1.5 (extreme)` on 91.3% and 89.5%. Even the fifth-ranked configuration (`Codex` with `Gemini 3.8 Flash`) reaches 1940.1 Elo.

Domain Knowledge vs. Statistical Feature Engineering. To understand the strategies the agents discover, we inspect the final pipelines from `Claude Code` with `Opus 5` and `Antigravity` with `Gemini 3.8 Flash` for all 51 TabArena datasets (Figure 3 and Appendix B.2 Table 6). We group them into three categories: domain-knowledge features, statistical and structural features, and context and calibration changes only. The agents modify the feature table in 95.1% of runs ($97/102$ across both configurations). They pair feature engineering (Φ_{feat}) with missing-value cleaning and target transforms (Φ_{clean}) on regression (3.11%–3.15% lower G-Mean root mean squared error, RMSE), and with minority-class context selection ($\mathcal{S}_{\mathrm{ctx}}$) and prior calibration (Ψ_{post}) on multiclass classification (7.64%–8.39% lower G-Mean log-loss). On the 17 datasets whose column names or task descriptions identify real-world quantities, such as physical, clinical, or economic variables (Cat. I), both agents write domain formulas directly from the schema (e.g., aerodynamic Strouhal numbers on `airfoil_self_noise`, 14.6%–17.3% lower test RMSE), reducing official test error across all folds by 7.3% (`Opus 5`) to 8.2% (`Gemini 3.8 Flash`). That is nearly three times the 2.3%–3.0% error reduction on the 34 datasets for `Opus 5` (29 for `Gemini 3.8 Flash`) with anonymized or generic schemas (Cat. II), where the agent relies on statistical transforms, frequency encodings, and graph degree features. Cat. I datasets account for 6 of the 10 largest test error reductions of each configuration. On the 5 runs where the agent does not change the feature table (Cat. III), 4 runs improve through context sampling in $\mathcal{S}_{\mathrm{ctx}}$ and/or prior calibration in Ψ_{post}, and the single run that tunes only hyperparameters is 0.30% worse.

Table 2: **Controlled ablation of `TabFM-Auto` vs. an unconstrained coding agent.** Both agents use the identical harness, model, sandbox, and a 6-hour search budget per dataset.

Method	Working Models	Overall Elo ↑	Cls. Elo ↑	Reg. Elo ↑
XGBoost (tuned + ensemble)	XGBoost	1360.4	1361.7	1428.7
LightGBM (tuned + ensemble)	LightGBM	1420.1	1419.6	1511.7
Coding Agent (AGY + Gemini 3.8 Flash)	Any ML models (GBDTs, PyTorch, etc.)	1468.8	1472.0	1615.3
AutoGluon 1.3 (4h)	Stacked GBDT & NN ensembles	1490.3	1473.4	1646.1
TabFM	TabFM (frozen)	1785.3	1768.7	2045.9
TabFM+	TabFM (frozen)	1856.0	1836.7	2169.2
TabFM-Auto (AGY + Gemini 3.8 Flash)	TabFM (frozen)	**1979.6**	**1937.5**	**2347.8**

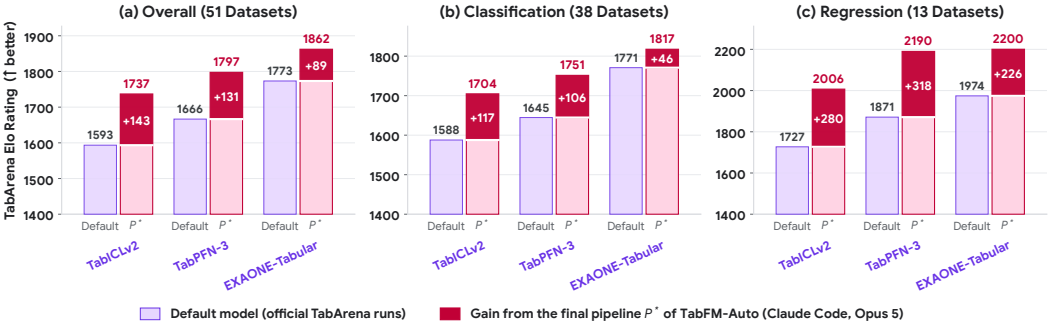

Figure 5: **Final pipelines found for `TabFM` also help other tabular foundation models.** *Default* is each model's official TabArena result, rated in the same fit as P^*. P^* runs the same frozen model within the final per-dataset pipelines that `TabFM-Auto` (`Claude Code`, `Opus 5`) found for `TabFM`, keeping only the context size from `TABFM_KWARGS` and without further search.

Validation-to-Test Generalization. To test whether iterative search overfits the 3-fold cross-validation split of fold 0, we evaluate up to 15 intermediate pipeline checkpoints per run from `Claude Code` with `Opus 5` and `Antigravity` with `Gemini 3.8 Flash`, taken at fixed fractions of the search budget, on the official test sets of all folds (Figure 4 and Appendix C Figures 14–15). We find that cross-validation gains on the training set of fold 0 (Figure 4a) carry over to the official test sets: mean test error falls by 4.0% (`Opus 5`) to 4.7% (`Gemini 3.8 Flash`) from P_0 to full budget (Figure 4b), and G-Mean error by 4.19% and 5.05% (Table 9). Both configurations score higher than `TabFM+` (1856 Elo) within the first 2.5% of the search (about 3 evaluations) and reach 1979–1993 Elo at full budget (Figure 4c). On the held-out test split of fold 0 alone (Table 5), `TabFM-Auto` also holds the #1 Overall, Classification, and Regression Elo ratings (Appendix B.1).

Ablation: Evolving Around Frozen `TabFM` vs. an Unconstrained MLE Agent. We compare `TabFM-Auto` against an unconstrained coding agent that uses the same `Antigravity` harness, `Gemini 3.8 Flash` model, and 6-hour budget, but may train and ensemble any machine learning models without `TabFM` (Table 2). `Coding Agent` outperforms individual tuned GBDTs by training and blending LightGBM, CatBoost, random forests, linear models, and neural networks (Appendix B.4, Figure 13). However, it is slightly worse than 4-hour `AutoGluon 1.3` and 510.8 Elo below `TabFM-Auto` (1468.8 vs. 1979.6): `TabFM`'s pretrained prior adds $+316.5$ Elo (to 1785.3) and pipeline search adds $+194.3$ Elo. This ablation supports our core claim that coding agents perform better when paired with a frozen foundation model, since model architectures and training hyperparameters form a wide search space and trained models vary from run to run.

Transfer to Other Tabular Foundation Models. The agents choose every pipeline by how well `TabFM` scores with it, which may make the final pipelines specific to `TabFM`. We test this by running the four functions of the final pipelines P^* from the three highest-rated configurations, unchanged and without further search, around three other frozen tabular foundation models (`TabPFN-3`, `TabICLv2`, and `EXAONE-Tabular`) in place of `TabFM` (Appendix B.3). From `TABFM_KWARGS`, we keep only the context size, as the other models do not support settings such as NNLS view weighting, feature crosses, and SVD features. The pipelines of all three configurations improve every model over

Table 3: **Cumulative evaluation count, API calls, token consumption, and tool invocation breakdown across 51 TabArena datasets.** (Top) Pipeline evaluations, LLM API requests, and prompt/generated token counts. (Bottom) Tool-call breakdown across the five action categories.

Agent Harness & Model	Datasets	# Evals	# API Calls	Prompt Tokens	Generated Tokens
Codex, Opus 5	51	1,257	18,948	1.95B	12.23M
Claude Code, Opus 5	51	4,296	7,452	1.28B	9.76M
Antigravity, Gemini 3.8 Flash	51	4,320	51,968	5.30B	20.50M
Claude Code, Gemini 3.8 Flash	51	4,602	38,401	3.75B	9.07M
Codex, Gemini 3.8 Flash	51	3,669	85,862	12.13B	20.25M

Agent Harness & Model	Total Calls	Shell Exec.	File Read	File Write	File Search	Task & Sched.
Codex, Opus 5	19,348	9,713 (50.2%)	270 (1.4%)	1,539 (8.0%)	120 (0.6%)	7,706 (39.8%)
Claude Code, Opus 5	7,381	6,385 (86.5%)	256 (3.5%)	690 (9.3%)	32 (0.4%)	18 (0.2%)
Antigravity, Gemini 3.8 Flash	49,352	25,467 (51.6%)	5,473 (11.1%)	8,688 (17.6%)	1,455 (2.9%)	8,269 (16.8%)
Claude Code, Gemini 3.8 Flash	38,063	16,288 (42.8%)	15,398 (40.5%)	5,743 (15.1%)	611 (1.6%)	23 (0.1%)
Codex, Gemini 3.8 Flash	85,389	32,630 (38.2%)	3,066 (3.6%)	3,963 (4.6%)	3,669 (4.3%)	42,061 (49.3%)

its default configuration, though by less than `TabFM` does. Those from `Claude Code` with `Opus 5` transfer best (Figure 5 and Appendix Figure 11), raising `TabICLv2` by +143.3 Elo, `TabPFN-3` by +130.8 Elo, and `EXAONE-Tabular` by +88.7 Elo. The discovered pipelines therefore carry over to other in-context learners.

Token and Tool Usage Across Harnesses and Models. Table 3 summarizes evaluation counts, token usage, and tool-call distributions across the five configurations. We observe that `Opus 5` and `Gemini 3.8 Flash` reach similar test error with different search styles: `Opus 5` uses more tokens per step for reasoning and evaluates roughly one-third as many candidate pipelines under `Codex`, whereas `Gemini 3.8 Flash` makes 4.4× to 5.2× more tool calls to test rapid, incremental edits. The harnesses differ in *Task & Sched.* calls because their shell tools either block until an evaluation finishes (`Claude Code`) or poll background processes (`Codex` and `Antigravity`).

4.2 MLE-Bench-Tabular

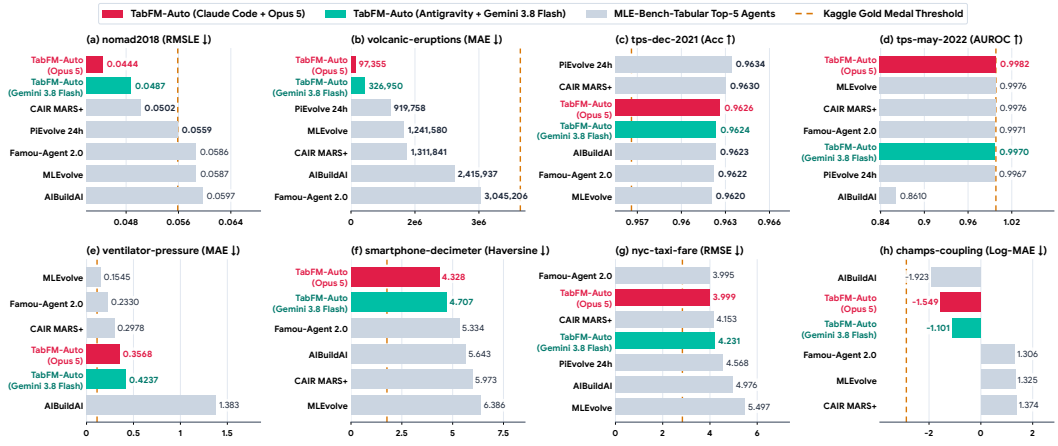

Figure 6: **Official test scores across all 8 MLE-Bench-Tabular competitions** comparing both `TabFM-Auto` configurations against five leading external agents (CAIR MARS+, PiEvolve 24h, MLEvolve, Famou-Agent 2.0, and AIBuildAI) and Kaggle Gold Medal thresholds.

Benchmark Definition and Auxiliary File Handling. We also evaluate `TabFM-Auto` on MLE-Bench-Tabular, the 8 tabular competitions of MLE-Bench (Chan et al., 2025) (Figure 6 and Appendix Table 8), on the official local test splits (using a 24-hour per-competition budget for `TabFM-Auto`

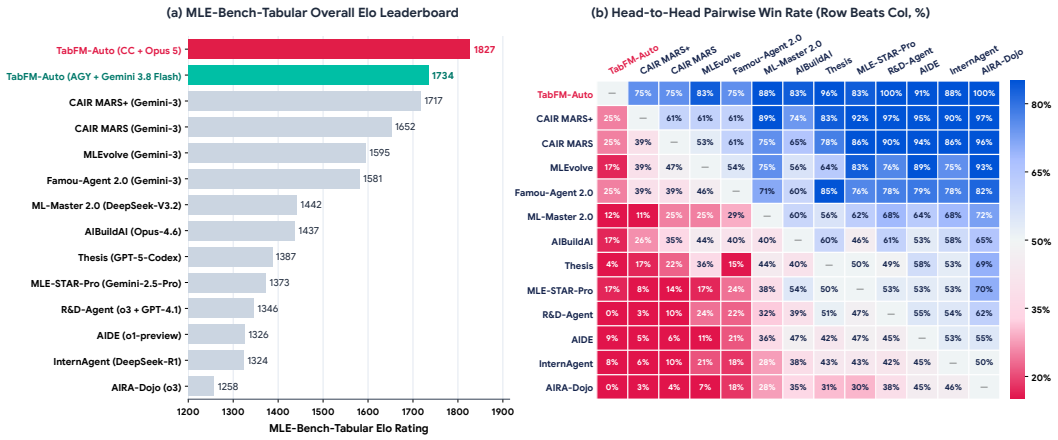

Figure 7: **MLE-Bench-Tabular overall comparison.** (a) Elo ratings of 14 MLE agents (our two configurations and all 12 external agents with complete 8-competition submissions, see Appendix D.1) and (b) pairwise win rates over the same run-vs-run matches showing the `Opus 5` variant.

against the official leaderboard submissions of external agents). In several competitions, the main signal is in auxiliary files such as seismic waveforms, satellite telemetry, or 3D molecular coordinates, which a single-table predictor cannot read directly. In Φ_{feat}, `TabFM-Auto` summarizes these files into tabular features (e.g., spectral energy or inter-atomic distances) and joins them to the main table before calling `TabFM` (Appendix D.1).

Comparison with End-to-End Agents on MLE-Bench-Tabular. External MLE-Bench agents, such as CAIR MARS+, MLEvolve, Famou-Agent 2.0, R&D-Agent, and AIDE, search over both data processing and task-specific predictive models, whereas `TabFM-Auto` adapts the data pipeline while keeping the predictor fixed. Although external agents lead on several individual competitions (Figure 6), `TabFM-Auto` achieves the highest aggregate Elo among the evaluated submissions. Across all graded pairwise matches against the 12 external MLE agents (Figure 7), `TabFM-Auto` ranks first and second overall (`Claude Code` with `Opus 5` at 1827 Elo, earning 4 Kaggle medals, and `Antigravity` with `Gemini 3.8 Flash` at 1734 Elo). `Opus 5` wins 86.7% of all graded matches, including 75%–88% against each of the five highest-rated external agents and 83%–100% against the rest (Figure 7b).

5 Conclusion and Discussion

We studied how a language model coding agent can evolve the data pipeline around a frozen tabular foundation model, combining language-model domain reasoning with fast forward-pass tabular inference to supply the semantics missing from synthetic pretraining. On all 51 TabArena datasets, our five `TabFM-Auto` configurations hold the top five overall positions (reaching 2013.0 Elo, +227.7 over `TabFM`). The discovered pipelines transfer without further search to other frozen tabular foundation models (+68.8 to +143.3 Elo). `TabFM-Auto` also ranks first overall on MLE-Bench-Tabular ahead of agents that train trees and neural networks from scratch.

Two limitations suggest directions for future work. Gains are generally smaller on anonymized tables, where the agent relies on statistical and relational features, and pipeline search requires repeated validation evaluations for each dataset. The discovered pipelines are short, readable programs. Operations collected across datasets could form a reusable library that warm-starts search on new tables with fewer evaluations. Incorporating the discovered formulas, cleaning rules, target transforms, and relational features into pretraining may lead to schema-aware tabular foundation models. Since `TabFM-Auto` already joins features derived from auxiliary files on MLE-Bench-Tabular, the same search could extend to multi-table relational databases.

AI USE STATEMENT

In accordance with the ICLR 2027 policy on the use of generative AI, we distinguish between the LLM coding agents that form the empirical subject of our experiments and the generative AI tools used by the authors during the research workflow. As the subject of study, `TabFM-Auto` evaluates LLM coding agents (`Claude Opus 5` and `Gemini 3.8 Flash` paired with the `Claude Code`, `Codex`, and `Antigravity` harnesses) on their ability to evolve data pipelines (`preprocess()`, `engineer()`, `sample()`, `postprocess()`, and `TABFM_KWARGS`) around frozen tabular foundation models in an isolated evaluation harness. Within the authors' own workflow, we used generative AI tools only to assist with coding and to edit manuscript prose for readability. We did not use generative AI for any other research tasks. All AI-assisted code and text were manually reviewed, tested, and verified for correctness by the authors. We take full responsibility for the final content of this work, including all text, code, claims, and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

The authors have read and adhere to the ICLR Code of Ethics. All experiments use publicly released, de-identified benchmark datasets (`TabArena` and `MLE-Bench`) in accordance with their open-source licenses, without human subjects, crowdworker annotation, or personally identifiable information. Because evaluating coding agents involves executing LLM-generated Python programs, the evaluation harness runs all agent-written code in the network-isolated Linux `bubblewrap` namespace sandbox described in Section 3 and Appendix A. This sandbox disables outbound network connections, mounts system libraries and benchmark data read-only, unmounts held-out test splits during search, and restricts writes to an ephemeral scratch directory. While automating data pipeline construction around tabular foundation models lowers the barrier to applying machine learning across scientific and enterprise tables, deploying evolved pipelines in high-stakes domains such as healthcare, finance, or public policy requires domain-specific fairness, calibration, and distribution-shift audits prior to real-world use.

REPRODUCIBILITY STATEMENT

All datasets and official train/test splits evaluated in this work come from two public benchmarks: the 51-dataset TabArena benchmark (Erickson et al., 2026) (3-fold cross-validation repeated 3 or 10 times, see Section 4.1) and the 8 tabular competitions of MLE-Bench (Chan et al., 2025) (official local splits, see Section 4.2 and Appendix D.1). Section 3 and Appendix A detail the pipeline search objective in Equation (1), the five pipeline components (`preprocess()`, `engineer()`, `sample()`, `postprocess()`, and `TABFM_KWARGS`), and the sandboxed evaluation protocol (including the identity baseline check P_0, filesystem split isolation, and deterministic row-order permutation). Section 4.1 and Appendix B.1–Appendix B.3 specify the Bradley–Terry Elo rating procedure (anchored to Random Forest at 1000 Elo across the 66-method TabArena pool), the single-fold and multi-fold evaluation protocols, the unconstrained coding-agent ablation (Appendix B.4), and the zero-shot pipeline transfer protocol across frozen tabular foundation models (`TabPFN-3`, `TabICLv2`, and `EXAONE-Tabular`). Sections 4.1 and 4.2 state the search budgets (up to 96 evaluations or 6 hours on one H100 GPU per TabArena dataset and 24 hours per MLE-Bench-Tabular competition), and Table 3 reports evaluation counts, token consumption, and tool-invocation distributions. Finally, Figure 8 lists the starting pipeline P_0, and Figure 3 and Table 6 provide representative evolved Python pipelines and stage-level taxonomy counts, while Tables 7–9 report complete transfer and per-dataset official test results across all 59 tasks.

REFERENCES

Ralph Allan Bradley and Milton E. Terry. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39:324, 1952.

Jun Shern Chan, Neil Chowdhury, Oliver Jaffe, James Aung, Dane Sherburn, Evan Mays, Giulio Starace, Kevin Liu, Leon Maksin, Tejal Patwardhan, Aleksander Madry, and Lilian Weng. MLE-bench: Evaluating machine learning agents on machine learning engineering. In *The Thirteenth*

International Conference on Learning Representations, 2025. URL https://openreview.net/forum?id=6s5uXNWGIh.

Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pp. 785–794, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450342322. doi: 10.1145/2939672.2939785. URL https://doi.org/10.1145/2939672.2939785.

Shangheng Du, Xiangchao Yan, Jinxin Shi, Zongsheng Cao, Shiyang Feng, Zichen Liang, Boyuan Sun, Tianshuo Peng, Yifan Zhou, Xin Li, Jie Zhou, Liang He, Bo Zhang, and Lei Bai. MLEvolve: A self-evolving framework for automated machine learning algorithm discovery, 2026. URL https://arxiv.org/abs/2606.06473.

Arpad E. Elo. *The Rating of Chessplayers, Past and Present*. Arco Publishing, 1978.

Nick Erickson, Jonas Mueller, Alexander Shirkov, Hang Zhang, Pedro Larroy, Mu Li, and Alexander Smola. Autogluon-tabular: Robust and accurate automl for structured data. *arXiv preprint arXiv:2003.06505*, 2020.

Nick Erickson, Lennart Purucker, Andrej Tschalzev, David Holzmüller, Prateek Mutalik Desai, David Salinas, and Frank Hutter. Tabarena: A living benchmark for machine learning on tabular data. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2026. URL https://openreview.net/forum?id=jZqCqpCLdU.

Matthias Feurer, Aaron Klein, Katharina Eggensperger, Jost Springenberg, Manuel Blum, and Frank Hutter. Efficient and robust automated machine learning. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015. URL https://proceedings.neurips.cc/paper_files/paper/2015/file/11d0e6287202fced83f79975ec59a3a6-Paper.pdf.

Deqing Fu, Tian-Qi Chen, Robin Jia, and Vatsal Sharan. Transformers learn to achieve second-order convergence rates for in-context linear regression. In *Advances in Neural Information Processing Systems*, volume 37, pp. 98675–98716. Curran Associates, Inc., 2024. doi: 10.52202/079017-3132. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/b2d4051f03a7038a2771dfbbe5c7b54e-Paper-Conference.pdf.

Deqing Fu, Tianyi Zhou, Mikhail Belkin, Vatsal Sharan, and Robin Jia. Convergent evolution: How different language models learn similar number representations. In *Third Conference on Language Modeling*, 2026. URL https://openreview.net/forum?id=RIoMmOkZ8E.

Anurag Garg, Muhammad Ali, Noah Hollmann, Lennart Purucker, Samuel Müller, and Frank Hutter. Real-tabpfn: Improving tabular foundation models via continued pre-training with real-world data, 2025. URL https://arxiv.org/abs/2507.03971.

Shivam Garg, Dimitris Tsipras, Percy S Liang, and Gregory Valiant. What can transformers learn in-context? a case study of simple function classes. In *Advances in Neural Information Processing Systems*, volume 35, pp. 30583–30598. Curran Associates, Inc., 2022. doi: 10.52202/068431-2217. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/c529dba08a146ea8d6cf715ae8930cbe-Paper-Conference.pdf.

Yury Gorishniy, Ivan Rubachev, Valentin Khrulkov, and Artem Babenko. Revisiting deep learning models for tabular data. In *Advances in Neural Information Processing Systems*, volume 34, pp. 18932–18943. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/9d86d83f925f2149e9edb0ac3b49229c-Paper.pdf.

Yury Gorishniy, Akim Kotelnikov, and Artem Babenko. Tabm: Advancing tabular deep learning with parameter-efficient ensembling. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=Sd4wYYOhmY.

Léo Grinsztajn, Klemens Flöge, Oscar Key, Felix Birkel, Philipp Jund, Brendan Roof, Benjamin Jäger, Dominik Safaric, Simone Alessi, Adrian Hayler, Mihir Manium, Rosen Yu, Felix Jablonski, Shi Bin Hoo, Anurag Garg, Jake Robertson, Magnus Bühler, Vladyslav Moroshan, Lennart Purucker, Clara Cornu, Lilly Charlotte Wehrhahn, Alessandro Bonetto, Bernhard Schölkopf, Sauraj Gambhir, Noah Hollmann, and Frank Hutter. Tabpfn-2.5: Advancing the state of the art in tabular foundation models, 2025. URL https://arxiv.org/abs/2511.08667.

Léo Grinsztajn, Klemens Flöge, Oscar Key, Felix Birkel, Philipp Jund, Brendan Roof, Mihir Manium, Shi Bin Hoo, Magnus Bühler, Anurag Garg, Dominik Safaric, Jake Robertson, Benjamin Jäger, Simone Alessi, Adrian Hayler, Vladyslav Moroshan, Lennart Purucker, Philipp Singer, Alan Arazi, Julien Siems, Jan Hendrik Metzen, Georg Grab, Nick Erickson, Siyuan Guo, Eliott Kalfon, Simon Bing, David Salinas, Clara Cornu, Lilly Charlotte Wehrhahn, Diana Kriuchkova, Kursat Kaya, Lydia Sidhoum, Marie Salmon, Jerry Chen, Madelon Hulsebos, Yann LeCun, Samuel Müller, Bernhard Schölkopf, Sauraj Gambhir, Noah Hollmann, and Frank Hutter. Tabpfn-3: Technical report, 2026. URL https://arxiv.org/abs/2605.13986.

Sungwon Han, Jinsung Yoon, Sercan O Arik, and Tomas Pfister. Large language models can automatically engineer features for few-shot tabular learning. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 17454–17479. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/han24f.html.

Stefan Hegselmann, Alejandro Buendia, Hunter Lang, Monica Agrawal, Xiaoyi Jiang, and David Sontag. Tabllm: Few-shot classification of tabular data with large language models. In *Proceedings of The 26th International Conference on Artificial Intelligence and Statistics*, volume 206 of *Proceedings of Machine Learning Research*, pp. 5549–5581. PMLR, 25–27 Apr 2023. URL https://proceedings.mlr.press/v206/hegselmann23a.html.

Noah Hollmann, Samuel Müller, Katharina Eggensperger, and Frank Hutter. TabPFN: A transformer that solves small tabular classification problems in a second. In *NeurIPS 2022 First Table Representation Workshop*, 2022. URL https://openreview.net/forum?id=eu9fVjVasr4.

Noah Hollmann, Samuel Müller, and Frank Hutter. Large language models for automated data science: Introducing CAAFE for context-aware automated feature engineering. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=9WSxQZ9mG7.

Noah Hollmann, Samuel Müller, Lennart Purucker, Arjun Krishnakumar, Max Körfer, Shi Bin Hoo, Robin Tibor Schirrmeister, and Frank Hutter. Accurate predictions on small data with a tabular foundation model. *Nature*, 637(8045):319–326, 2025.

Rasa Hosseinzadeh, Alex Labach, Zexin Xue, Shuyi Han, Valentin Thomas, and Anthony L. Caterini. Tabdpt-turbo: Efficient in-context learning for tabular prediction, 2026. URL https://arxiv.org/abs/2608.01400.

Qian Huang, Jian Vora, Percy Liang, and Jure Leskovec. MLAgentBench: Evaluating language agents on machine learning experimentation. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 20271–20309. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/huang24y.html.

David R. Hunter. MM algorithms for generalized Bradley–Terry models. *The Annals of Statistics*, 32 (1):384–406, 2004.

Zhengyao Jiang, Dominik Schmidt, Dhruv Srikanth, Dixing Xu, Ian Kaplan, Deniss Jacenko, and Yuxiang Wu. AIDE: Ai-driven exploration in the space of code, 2025. URL https://arxiv.org/abs/2502.13138.

Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/6449f44a102fde848669bdd9eb6b76fa-Paper.pdf.

Myung Jun Kim, Leo Grinsztajn, and Gael Varoquaux. CARTE: Pretraining and transfer for tabular learning. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 23843–23866. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/kim24d.html.

Weihao Kong, Erez Louidor Ilan, Shuxin Nie, Taman Narayan, Rajat Sen, Yichen Zhou, Deqing Fu, Samet Oymak, and Abhimanyu Das. TabFM: A zero-shot foundation model for tabular data. *Google Research Blog*, 2026. URL https://research.google/blog/introducing-tabfm-a-zero-shot-foundation-model-for-tabular-data/.

Charles L. Lawson and Richard J. Hanson. *Solving Least Squares Problems*. Society for Industrial and Applied Mathematics, 1995. doi: 10.1137/1.9781611971217. URL https://epubs.siam.org/doi/abs/10.1137/1.9781611971217.

Yingcong Li, Muhammed Emrullah Ildiz, Dimitris Papailiopoulos, and Samet Oymak. Transformers as algorithms: Generalization and stability in in-context learning. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 19565–19594. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/li23l.html.

Junwei Ma, Valentin Thomas, Rasa Hosseinzadeh, Alex Labach, Jesse C. Cresswell, Keyvan Golestan, Guangwei Yu, Anthony L. Caterini, and Maksims Volkovs. TabDPT: Scaling tabular foundation models on real data. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=pIZxEOZCId.

Samuel Müller, Noah Hollmann, Sebastian Pineda Arango, Josif Grabocka, and Frank Hutter. Transformers can do bayesian inference. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=KSugKcbNf9.

Jaehyun Nam, Kyuyoung Kim, Seunghyuk Oh, Jihoon Tack, Jaehyung Kim, and Jinwoo Shin. Optimized feature generation for tabular data via LLMs with decision tree reasoning. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=APSBwuMopO.

Alexander Novikov, Ngân Vũ, Marvin Eisenberger, Emilien Dupont, Po-Sen Huang, Adam Zsolt Wagner, Sergey Shirobokov, Borislav Kozlovskii, Francisco J. R. Ruiz, Abbas Mehrabian, M. Pawan Kumar, Abigail See, Swarat Chaudhuri, George Holland, Alex Davies, Sebastian Nowozin, Pushmeet Kohli, and Matej Balog. AlphaEvolve: A coding agent for scientific and algorithmic discovery, 2025. URL https://arxiv.org/abs/2506.13131.

John Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *Advances in Large Margin Classifiers*. MIT Press, 1999.

Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. Catboost: unbiased boosting with categorical features. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, pp. 6639–6649, Red Hook, NY, USA, 2018. Curran Associates Inc.

Zhenting Qi, Huangyuan Su, Ao Qu, Chenyu Wang, Yu Yao, Han Zheng, Kushal Chattopadhyay, Guowei Xu, Zihan Wang, Weirui Ye, Vijay Janapa Reddi, Ju Li, Paul Pu Liang, Himabindu Lakkaraju, Sham Kakade, and Yilun Du. Economy of minds: Emerging multi-agent intelligence with economic interactions, 2026. URL https://arxiv.org/abs/2606.02859.

Jingang Qu, David Holzmüller, Gaël Varoquaux, and Marine Le Morvan. TabICL: A tabular foundation model for in-context learning on large data. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=0VvD1PmNzM.

Jingang Qu, David Holzmüller, Gaël Varoquaux, and Marine Le Morvan. TabICLv2: A better, faster, scalable, and open tabular foundation model. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=SxsyLjIfWB.

Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M. Pawan Kumar, Emilien Dupont, Francisco J. R. Ruiz, Jordan S. Ellenberg, Pengming Wang, Omar Fawzi, Pushmeet Kohli, and Alhussein Fawzi. Mathematical discoveries from program search with large language models. *Nature*, 625(7995):468–475, 2024. doi: 10.1038/s41586-023-06924-6. URL https://doi.org/10.1038/s41586-023-06924-6.

Mustafa Tajjar, Alexander Pfefferle, Lennart Purucker, and Frank Hutter. Towards pretraining text encoders for tabpfn, 2026. URL https://arxiv.org/abs/2606.04876.

Alexander Tornede, Difan Deng, Theresa Eimer, Joseph Giovanelli, Aditya Mohan, Tim Ruhkopf, Sarah Segel, Daphne Theodorakopoulos, Tanja Tornede, Henning Wachsmuth, and Marius Lindauer. AutoML in the age of large language models: Current challenges, future opportunities and risks. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=cAthubStyG.

Johannes Von Oswald, Eyvind Niklasson, Ettore Randazzo, Joao Sacramento, Alexander Mordvintsev, Andrey Zhmoginov, and Max Vladymyrov. Transformers learn in-context by gradient descent. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 35151–35174. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/von-oswald23a.html.

Guowei Xu, Zhenting Qi, Huangyuan Su, Weirui Ye, Himabindu Lakkaraju, Sham M. Kakade, and Yilun Du. Self-improving language models with bidirectional evolutionary search, 2026. URL https://arxiv.org/abs/2605.28814.

Jiahuan Yan, Bo Zheng, Hongxia Xu, Yiheng Zhu, Danny Chen, Jimeng Sun, Jian Wu, and Jintai Chen. Making pre-trained language models great on tabular prediction. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=anzIzGZuLi.

Xu Yang, Xiao Yang, Shikai Fang, Yifei Zhang, Jian Wang, Bowen Xian, Qizheng Li, Jingyuan Li, Minrui Xu, Yuante Li, Haoran Pan, Yuge Zhang, Weiqing Liu, Yelong Shen, Weizhu Chen, and Jiang Bian. R&d-agent: An llm-agent framework towards autonomous data science, 2025. URL https://arxiv.org/abs/2505.14738.

Tianyi Zhou, Deqing Fu, Vatsal Sharan, and Robin Jia. Pre-trained large language models use fourier features to compute addition. In *Advances in Neural Information Processing Systems*, volume 37, pp. 25120–25151. Curran Associates, Inc., 2024. doi: 10.52202/079017-0792. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/2cc8dc30e52798b27d37b795cc153310-Paper-Conference.pdf.

Tianyi Zhou, Deqing Fu, Mahdi Soltanolkotabi, Robin Jia, and Vatsal Sharan. FoNE: Precise single-token number embeddings via fourier features. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=g0vtWmwDDh.

APPENDIX - TABLE OF CONTENTS

A Verification Protocol and Test Isolation **15**

B Full TabArena Results and Taxonomy Analysis **16**

B.1 Overall Leaderboard and Baseline Comparison 16

B.2 Analysis of Discovered Pipelines 19

B.3 Transferring Final Pipelines to Other Tabular Foundation Models 21

B.4 Models Used by the Unconstrained Coding Agent 22

C Search Dynamics and Generalization Analysis **23**

D MLE-Bench-Tabular Analysis and Per-Competition Results **24**

D.1 Competition-by-Competition Analysis on MLE-Bench-Tabular 24

E Per-Dataset Official Test Scores on TabArena **26**

A VERIFICATION PROTOCOL AND TEST ISOLATION

The evaluation harness enforces three checks during search and testing so that the error reductions reported for `TabFM-Auto` come from the discovered data pipeline rather than from reading held-out test splits or from replacing `TabFM` with another model.

Identity Baseline and Frozen-Model Check. Every search run begins by evaluating the identity pass-through pipeline $P_0 = (\mathrm{Id}, \mathrm{Id}, \mathrm{Id}, \mathrm{Id})$ (Figure 8) as its first evaluation (step 1). P_0 passes the unmodified table to `TabFM` with default constructor settings on the same 3-fold cross-validation splits of fold 0's training set, and its score $\mathcal{U}_{\text{val}}(P_0)$ sets the starting threshold: the agent keeps only edits that improve the validation score over P_0. The harness runs the PyTorch checkpoint of `TabFM` with a fixed ensemble of 8 members for every pipeline, so the test errors of P_0 differ slightly from the published `TabFM` results in Table 9. The harness, not the pipeline, calls the frozen 400M-parameter `TabFM` model f_{θ^*} on each candidate's transformed table and passes its predictions to `postprocess()`, so every final prediction goes through f_{θ^*}.

Operating-System Sandboxing and Split Isolation. Each TabArena dataset defines 9 or 30 official train/test evaluation splits (3-fold cross-validation repeated 3 or 10 times). During pipeline search, the evaluation harness gives the agent access only to fold 0's training split, partitioning it into 3 internal cross-validation folds $(\mathcal{D}_{\text{subtrain}}, \mathcal{D}_{\text{val}})$ to compute $\mathcal{U}_{\text{val}}$. The harness runs every candidate pipeline in an unprivileged bubblewrap Linux namespace sandbox that disables outbound network access, mounts system libraries read-only, and restricts writes to an ephemeral scratch directory. The sandbox also unmounts fold 0's official test split along with the train and test splits of all other folds across the 51 datasets. Only after search finishes and P^* is frozen does the harness score P^* once on those held-out splits in an isolated environment.

Row-Order Permutation Against Index Leakage. The raw files of several public classification datasets are sorted by target class or acquisition timestamp, so a pipeline that preserves the original row order could infer labels from row positions. The harness therefore deterministically shuffles all classification folds at load time and restores the original order only when scoring final predictions.

```python
import numpy as np

TABFM_KWARGS = {}  # TabFM constructor settings; {} keeps the defaults

def preprocess(X_train, y_train, X_test):          # Φ_clean
  return X_train, y_train, X_test

def engineer(X_train, y_train, X_test):            # Φ_feat
  return X_train, X_test

def sample(X_train, y_train, X_test, max_rows):    # S_ctx
  return [np.arange(len(X_train))]  # one context view with all training rows

def postprocess(pred, y_train_orig, X_test):       # Ψ_post
  return pred
```

Figure 8: **Identity pipeline P_0 that starts every search run** (the starter `pipeline.py`, simplified). Every function returns its inputs unchanged and `TABFM_KWARGS` is empty. The first evaluation thus scores `TabFM` with default constructor settings on the raw table. The harness fits the frozen `TabFM` (8 ensemble members, configured by `TABFM_KWARGS`) once per context view returned by `sample()`, averages the test predictions over views, and passes them to `postprocess()`, where `y_train_orig` is the untransformed training target used to invert target transforms. The hooks also receive a `row_id` column, which the harness drops before calling `TabFM`. `sample()` is optional, and the listing writes out its default. `engineer()` may return at most 500 columns, so P_0 fails on the three datasets with more columns (`Bioresponse`, `hiva_agnostic`, and `QSAR-TID-11`).

Table 4: **TabArena overall leaderboard across all 51 datasets.** Super/subscripts give the 95% bootstrap percentile interval of each Elo rating relative to its median over 100 resampling rounds, and are asymmetric because the Bradley–Terry bootstrap distribution is right-skewed.

#	Method	Elo ↑	Wins ↑	Improv. (%) ↓	G-Mean ↓
1	TabFM-Auto (Codex, Opus 5)	**2013.0** $^{+123.7}_{-103.2}$	25.00	1.42%	0.3400
2	TabFM-Auto (Claude Code, Opus 5)	1993.6 $^{+126.8}_{-124.3}$	25.47	**1.26%**	0.3402
3	TabFM-Auto (Antigravity, Gemini 3.8 Flash)	1979.6 $^{+111.3}_{-95.7}$	**26.12**	1.82%	**0.3371**
4	TabFM-Auto (Claude Code, Gemini 3.8 Flash)	1957.8 $^{+106.1}_{-103.7}$	22.67	1.88%	0.3446
5	TabFM-Auto (Codex, Gemini 3.8 Flash)	1940.1 $^{+120.1}_{-133.9}$	25.06	1.81%	0.3435
6	TabFM+	1856.0 $^{+118.4}_{-102.7}$	17.58	2.20%	0.3513
7	TabFM	1785.3 $^{+114.9}_{-125.9}$	9.74	3.20%	0.3550
8	EXAONE-Tabular	1764.6 $^{+90.9}_{-72.4}$	5.56	6.05%	0.3703
9	AutoGluon 1.5 (extreme)	1668.4 $^{+80.6}_{-79.2}$	2.11	6.79%	0.3718
10	TabPFN-3	1660.1 $^{+84.0}_{-69.6}$	1.61	8.14%	0.3811
11	AutoGluon 1.4 (4h)	1612.3 $^{+73.0}_{-61.3}$	0.58	9.50%	0.3913
12	TabPFN-2.6	1604.8 $^{+71.3}_{-73.9}$	0.28	9.69%	0.3896
13	TabICLv2	1590.1 $^{+114.1}_{-87.9}$	1.92	9.04%	0.3856
14	RealTabPFN-2.5	1521.9 $^{+92.8}_{-65.7}$	0.10	10.38%	0.3918
15	AutoGluon 1.3 (4h)	1490.3 $^{+67.1}_{-79.6}$	0.18	12.15%	0.4115
16	TabDPT-Turbo	1455.8 $^{+70.0}_{-71.3}$	0.67	13.11%	0.4127

B Full TabArena Results and Taxonomy Analysis

B.1 Overall Leaderboard and Baseline Comparison

Table 4 and Figure 9 report the full leaderboard across all 51 TabArena datasets, comparing our five `TabFM-Auto` configurations against tuned GBDTs, deep tabular models, 4-hour `AutoGluon 1.5`, and other tabular foundation models. `TabFM` and `TabFM+` already outperform individual

Table 5: **TabArena Fold-0 held-out test Elo ratings across Overall (51 datasets), Classification (38 datasets), and Regression (13 datasets).** Every method is scored on the official held-out test split of Fold 0, which shares no rows with Fold 0's training split used for 3-fold cross-validation to select P^*. As in Table 4, each `TabFM-Auto` configuration is rated individually against the 66-method TabArena pool using bootstrap Bradley–Terry estimation anchored to `RandomForest (default)` $= 1000$. Super/subscripts give the 95% bootstrap percentile interval of each rating relative to its median.

#	Method	Overall Elo (51) ↑	Classification Elo (38) ↑	Regression Elo (13) ↑
1	`TabFM-Auto (Codex, Opus 5)`	**1950.7** $^{+190.3}_{-169.3}$	**1877.5** $^{+197.0}_{-183.6}$	2616.3 $^{+355.1}_{-188.8}$
2	`TabFM-Auto (Claude Code, Gemini 3.8 Flash)`	1862.8 $^{+181.0}_{-163.7}$	1805.0 $^{+186.1}_{-187.6}$	2336.1 $^{+505.0}_{-318.9}$
3	`TabFM+`	1853.2 $^{+194.2}_{-177.3}$	1782.8 $^{+204.9}_{-187.4}$	2409.9 $^{+294.5}_{-173.8}$
4	`TabFM-Auto (Claude Code, Opus 5)`	1843.3 $^{+202.8}_{-172.0}$	1763.6 $^{+203.7}_{-184.8}$	2562.2 $^{+422.8}_{-217.6}$
5	`TabFM-Auto (Antigravity, Gemini 3.8 Flash)`	1791.9 $^{+181.7}_{-155.4}$	1710.5 $^{+201.8}_{-154.3}$	2532.4 $^{+319.5}_{-206.1}$
6	`TabFM`	1790.2 $^{+181.4}_{-168.5}$	1733.0 $^{+201.1}_{-190.7}$	2227.2 $^{+262.9}_{-144.8}$
7	`TabFM-Auto (Codex, Gemini 3.8 Flash)`	1785.6 $^{+239.5}_{-187.4}$	1703.2 $^{+224.5}_{-193.9}$	**2645.6** $^{+353.7}_{-184.3}$
8	`EXAONE-Tabular`	1772.2 $^{+133.7}_{-128.9}$	1740.4 $^{+169.0}_{-140.8}$	2032.4 $^{+211.4}_{-180.0}$
9	`AutoGluon 1.5` (extreme, 4h)	1661.0 $^{+136.1}_{-136.4}$	1612.8 $^{+148.7}_{-160.1}$	2019.1 $^{+126.6}_{-81.3}$
10	TabPFN-3	1653.7 $^{+125.6}_{-112.5}$	1608.0 $^{+135.8}_{-126.8}$	1971.0 $^{+294.0}_{-186.6}$
11	TabPFN-2.6	1626.9 $^{+110.6}_{-112.8}$	1582.3 $^{+125.9}_{-118.0}$	1916.7 $^{+195.7}_{-121.8}$
12	AutoGluon 1.4 (4h)	1596.2 $^{+121.0}_{-123.7}$	1568.9 $^{+143.2}_{-151.1}$	1800.7 $^{+146.1}_{-99.7}$
13	TabICLv2	1589.9 $^{+132.3}_{-122.9}$	1573.8 $^{+159.0}_{-136.8}$	1742.0 $^{+355.4}_{-233.9}$
14	RealTabPFN-2.5	1540.3 $^{+104.1}_{-97.9}$	1525.2 $^{+121.8}_{-107.2}$	1685.2 $^{+248.6}_{-220.9}$
15	AutoGluon 1.3 (4h)	1505.9 $^{+94.4}_{-93.4}$	1473.6 $^{+116.9}_{-106.2}$	1720.6 $^{+125.4}_{-114.2}$
16	Mitra	1452.5 $^{+121.5}_{-119.0}$	1439.5 $^{+144.7}_{-139.3}$	1586.7 $^{+205.4}_{-171.4}$
17	RealMLP (tuned + ensemble)	1449.9 $^{+88.8}_{-92.5}$	1418.1 $^{+103.0}_{-115.4}$	1665.7 $^{+114.1}_{-87.0}$
18	TabDPT-Turbo	1418.0 $^{+94.7}_{-83.6}$	1373.5 $^{+110.9}_{-102.8}$	1697.1 $^{+146.7}_{-108.2}$
19	LightGBM (tuned + ensemble)	1392.8 $^{+83.7}_{-79.8}$	1366.9 $^{+107.3}_{-93.2}$	1566.8 $^{+96.0}_{-83.2}$
20	TabM (tuned + ensemble)	1387.6 $^{+103.7}_{-103.0}$	1372.7 $^{+135.5}_{-129.6}$	1502.7 $^{+145.5}_{-118.9}$
21	CatBoost (tuned + ensemble)	1379.0 $^{+89.9}_{-95.7}$	1356.3 $^{+109.7}_{-116.8}$	1523.6 $^{+124.7}_{-89.1}$

Table 6: **Taxonomy of discovered pipelines on TabArena.** Each final pipeline P^* from `Claude Code` with `Opus 5` and `Antigravity` with `Gemini 3.8 Flash` across the 51 datasets is classified into one of three mutually exclusive categories. *Configuration* lists each agent setup as well as *Combined Best* (selecting the pipeline with the higher 3-fold cross-validation score $\mathcal{U}_{\text{val}}$ between the two configurations per dataset). *Datasets* reports the fraction of the 51 TabArena datasets assigned to each category, and *Mean* and *Median* report the mean and median relative official test error reduction ($\Delta\%$) over `TabFM` across datasets in that category (RMSE for regression, $1 - \text{AUROC}$ for binary classification, and log-loss for multiclass classification).

Category	Representative Features Built in Code	Configuration	Datasets	Mean ($\Delta\%$)	Median ($\Delta\%$)
I: Domain-Knowledge Features	**Physics & Engineering**: Strouhal $St = f\delta^*/U_\infty$, Helmholtz He (`airfoil`); water-to-binder ratio (`concrete`)	Claude Code, Opus 5	17 / 51	**+7.25%**	**+3.01%**
	Clinical Medicine: ICD-9 organ chapters (`Diabetes130US`); MAP $=$ (2DBP + SBP)/3, Shock Index (`MIC`)	Antigravity, Gemini 3.8 Flash	17 / 51	**+8.16%**	**+2.74%**
	Science & Genomics: Coil `width` $= 0 \rightarrow$ NaN (`anneal`); SDSS color indices (`SDSS17`); DNA splice motifs (`splice`)	*Combined Best*	*17 / 51*	***+8.85%***	***+3.01%***
II: Statistical & Structural Features	**Relational Graph Degrees**: Marginal entity counts, pairwise co-occurrences, distinct-B-per-A (`Amazon_employee`)	Claude Code, Opus 5	34 / 51	+2.31%	+0.90%
	Frequency & Missingness: Level counts, cross-fitted target means, missingness-aware column ranking (`kddcup09_appetency`)	Antigravity, Gemini 3.8 Flash	29 / 51	+2.99%	+1.32%
	Wide-Table Compression: Collinearity pruning and truncated SVD projections (`Bioresponse`, `hiva_agnostic`)	*Combined Best*	*34 / 51*	*+3.87%*	*+1.71%*
III: Context & Calibration Only	**Context Selection**: Minority-class context oversampling (`sample()`) without modifying $\mathbf{X}$	Claude Code, Opus 5	0 / 51	—	—
	Prior Calibration: Log-odds prior alignment (`postprocess()`) and temperature/vector scaling (`TABFM_KWARGS`)	Antigravity, Gemini 3.8 Flash	5 / 51	+2.16%	+0.19%

GBDT families and 4-hour `AutoGluon 1.5` (1668.4 Elo). Evolving the data pipeline around frozen `TabFM` reduces overall oracle improvability from 3.20% (`TabFM`) and 2.20% (`TabFM+`) down to 1.26%–1.88% and more than doubles the number of dataset wins (from 9.74 for `TabFM` to 26.12 for `Antigravity` with `Gemini 3.8 Flash` and 25.00 for `Codex` with `Opus 5`).

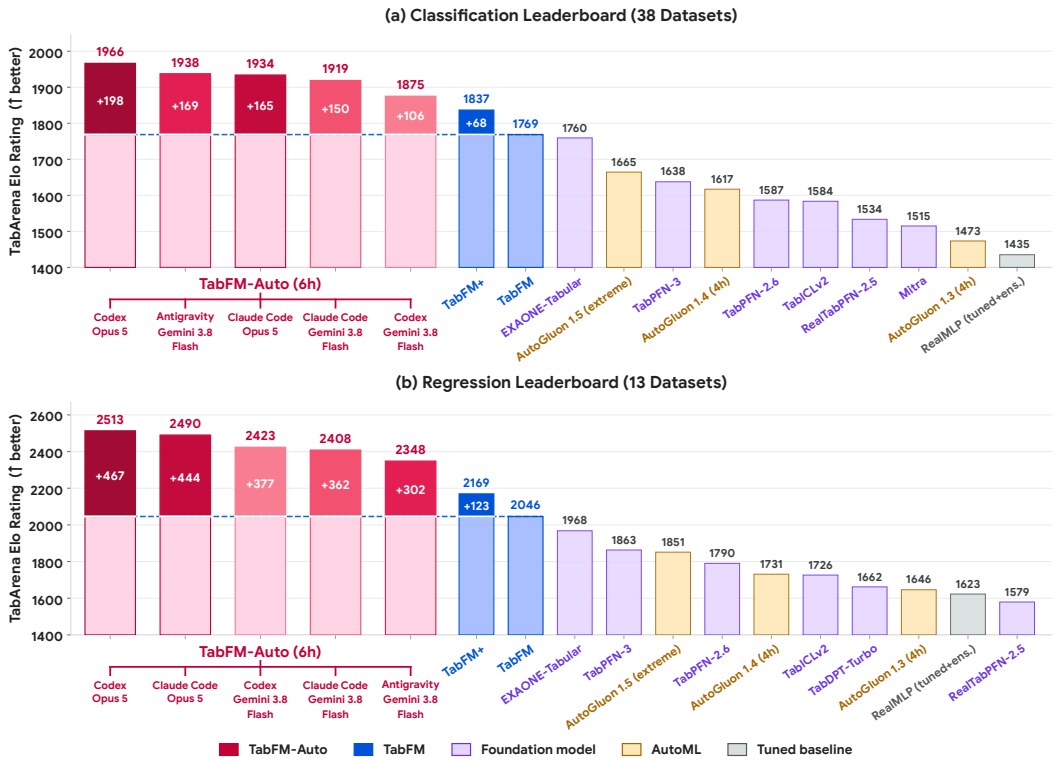

Figure 9: **TabArena Elo ratings, computed separately on the 38 classification (top) and 13 regression (bottom) datasets.** Red bars denote `TabFM-Auto` configurations (#1–#5), while blue bars denote `TabFM+` (#6) and `TabFM` (#7).

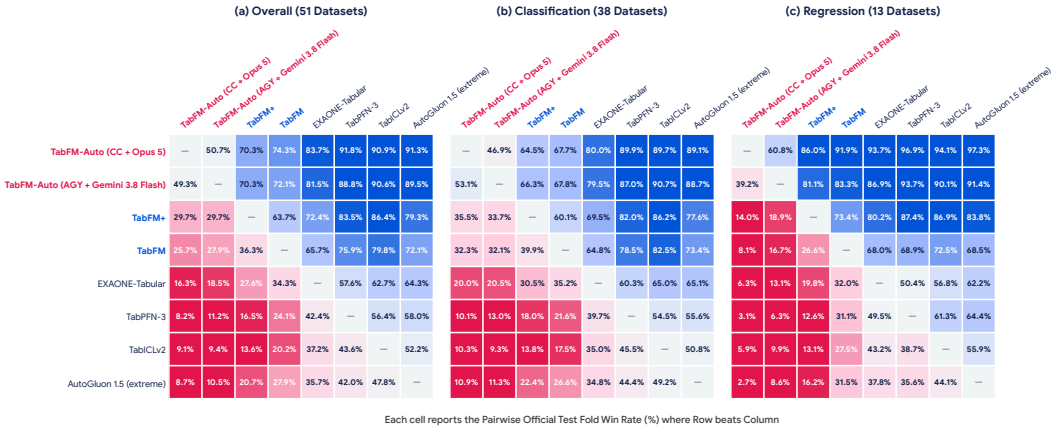

Figure 10: **Pairwise official test fold win rates on TabArena** across (a) all 51 datasets, (b) 38 classification datasets, and (c) 13 regression datasets. Each cell reports the percentage of dataset-fold instances where the row method achieves lower official test error than the column method.

Fold-0 Held-Out Evaluation and Cross-Fold Pipeline Transfer. `TabFM-Auto` searches for a single Python data pipeline script $P^* = (\Phi_{\text{clean}}, \Phi_{\text{feat}}, \mathcal{S}_{\text{ctx}}, \Psi_{\text{post}})$ per dataset using 3-fold cross-validation on Fold 0's training split $\mathcal{D}^{(0)}_{\text{train}}$ and evaluates P^* across all $S \in \{9, 30\}$ official evaluation folds $k \in \{0, \ldots, S-1\}$. Because TabArena repeats 3-fold cross-validation, the test sets $\mathbf{X}^{(k)}_{\text{test}}$ of Folds $k \geq 1$ share rows with $\mathcal{D}^{(0)}_{\text{train}}$, so selecting P^* on $\mathcal{D}^{(0)}_{\text{train}}$ could inflate their test scores. Three aspects of our protocol and results are relevant here:

1. *One search per dataset:* As in manual feature engineering, the agent designs a single dataset-level pipeline (which domain ratios to compute, which sentinel values to mask, and whether to transform the target) and reuses it across every train/test split, rather than writing 9 or 30 different feature sets for the same dataset. Running an independent 6-hour agent search on all 816 folds across five configurations (4,080 runs) would increase LLM token and GPU compute costs by 16× (from 1.28B–12.13B prompt tokens per 51-dataset sweep in Table 3 to 20.5B–194.1B tokens). The five sweeps in Table 3 cost about $17.6K in LLM API fees at Vertex AI list prices. This protocol lets the search see rows that later serve as test rows in Folds $k \geq 1$, so Fold 0 is the only fully held-out split.
2. *Symbolic program structure vs. fitted numerical parameters:* What transfers across folds is only a short Python source script (`pipeline.py`) specifying column-level operations, such as physical scaling laws ($St = f\delta^*/U_\infty$), sentinel rules (`width` $= 0 \rightarrow$ NaN), and target link functions ($\log(1+y)$). On every evaluation fold $k \in \{0, \ldots, S-1\}$, the harness executes `pipeline.py` in a fresh isolated process where all data-dependent statistics (imputers, frequency maps, SVD bases, empirical class priors) and `TabFM`'s in-context conditioning are fitted only on fold k's training split $(\mathbf{X}^{(k)}_{\text{train}}, \mathbf{y}^{(k)}_{\text{train}})$ before predicting on fold k's held-out test split $\mathbf{X}^{(k)}_{\text{test}}$. No model weights, fitted statistics, or row predictions from $\mathcal{D}^{(0)}_{\text{train}}$ carry over: $\mathcal{D}^{(0)}_{\text{train}}$ affects Folds $k \geq 1$ only through which operations and settings the agent kept.
3. *Fold-0 held-out evaluation (Table 5):* Scored on Fold 0's test split $\mathcal{D}^{(0)}_{\text{test}}$ alone, which shares no rows with $\mathcal{D}^{(0)}_{\text{train}}$, `TabFM-Auto` holds the #1 Overall and #1 Classification ratings (1950.7 and 1877.5 Elo, both `Codex` with `Opus 5`) and the #1–#4 Regression ratings (2532.4–2645.6 Elo, led by `Codex` with `Gemini 3.8 Flash`), and all five configurations outrank `EXAONE-Tabular` (1772.2) and 4-hour `AutoGluon 1.5` (1661.0) overall. For the five configurations, pairwise win rates against the 62 TabArena methods with complete results are 91.1%–95.7% on Fold 0 versus 94.4%–95.9% on Folds $k \geq 1$ (95.7% vs. 95.9% for `Codex` with `Opus 5`, 93.6% vs. 94.7% for `Claude Code` with `Gemini 3.8 Flash`, and 93.0% vs. 95.8% for `Claude Code` with `Opus 5`). For `Codex` with `Opus 5`, the top-rated configuration in Table 4, the G-Mean test error reduction over `TabFM` on Fold 0 is close to that on Folds $k \geq 1$ (3.74% vs. 4.53% on classification and 3.23% vs. 3.28% on regression).

B.2 Analysis of Discovered Pipelines

We analyze the final pipelines $P^* = (\Phi_{\text{clean}}, \Phi_{\text{feat}}, \mathcal{S}_{\text{ctx}}, \Psi_{\text{post}})$ discovered by `Claude Code` with `Opus 5` and `Antigravity` with `Gemini 3.8 Flash` across all 51 TabArena datasets (Table 6). We count how often the agents edit each stage and inspect the functions they write to see how they address the limitations of synthetic pretraining.

Data Cleaning and Target Conditioning (Φ_{clean}). Both `Opus 5` and `Gemini 3.8 Flash` modify the cleaning and target-conditioning stage on 33.3% of the benchmark (17/51 datasets each). Because `TabFM` embeds continuous columns through shared linear projections before computing row-wise attention, uncleaned numerical placeholders (such as `0` for unmeasured steel coil width on `anneal` or `-1` and `999` on administrative records) distort row-wise attention weights. By checking column names and value histograms, the agents replace these placeholders with missing-value indicators. On regression datasets with skewed targets, Φ_{clean} also applies reversible transforms $g(\mathbf{y})$ such as $\log(1+y)$ or Box–Cox scaling before calling `TabFM`, and Ψ_{post} inverts them with g^{-1} at the output.

I: Domain-Knowledge Features (Φ_{feat} on Informative Schemas, 17 Datasets). `Opus 5` modifies the feature table via Φ_{feat} (or Φ_{clean}) on 100% of datasets (51/51) and `Gemini 3.8 Flash` on 90.2% (46/51), for a pooled rate of 95.1% (97/102). On the 17 datasets whose column names or task descriptions identify real-world quantities, such as physical, clinical, engineering, or economic variables (Category I in Table 6), both models translate domain relationships directly into nonlinear composite features, reducing multi-fold official test error by +7.25% (`Opus 5`) and +8.16% (`Gemini 3.8 Flash`), and by +8.85% under validation-selected *Combined Best*. These 17 datasets account for 6 of the 10 largest dataset improvements on TabArena for each configuration. These domain features fall into three groups where explicit metadata-guided features complement synthetic pretraining:

- *Physical Ratios and Engineering Formulas:* Alternating row-and-column attention approximates smooth additive combinations well, but benefits from explicit multiplicative ratios and trigonometric projections on physical tasks. On engineering datasets, the agents build dimensionless scaling terms directly from variable names (Figure 3-I). Examples include Strouhal numbers $St = f\delta^*/U_\infty$ and Helmholtz compactness $He = fc/a_0$ on `airfoil_self_noise` ($+14.6\%$ and $+17.3\%$ test RMSE reduction for `Opus 5` and `Gemini 3.8 Flash`, respectively), water-to-binder ratios $W/(C + 0.8S + 0.3F)$ on `concrete_strength` ($+3.8\%$, `Opus 5`), and mass-weighted thermal conductivity ratios on `superconductivity`.
- *Clinical Diagnostic Codes and Vital-Sign Ratios:* On medical records where categorical columns store alphanumeric diagnostic codes, a tabular foundation model treats each code as an unrelated discrete token. On `Diabetes130US`, the agents parse 3-digit ICD-9 strings into 19 physiological organ-system chapters with dedicated offsets for supplementary V-codes and E-codes, coarsening hundreds of sparse categories into clinically meaningful strata. On intensive-care and obstetric cohorts (`MIC` and `maternal_health_risk`), the agents combine systolic blood pressure (SBP), diastolic blood pressure (DBP), and heart rate (HR) into standard clinical indices (Mean Arterial Pressure $\mathrm{MAP} = (2\mathrm{DBP} + \mathrm{SBP})/3$, Pulse Pressure $\mathrm{SBP} - \mathrm{DBP}$, and Shock Index HR/SBP).
- *Domain Sequence Motifs and Spectral Color Indices:* On scientific tables encoding sequences or multi-band fluxes, the agents construct domain-specific local features: extracting position-specific dinucleotide and trinucleotide sequence motifs around the donor/acceptor junction on `splice` ($+22.2\%$ test log-loss reduction, `Opus 5`) and forming pairwise photometric color indices ($u-g, g-r, r-i, i-z$) across adjacent filter bands on Sloan Digital Sky Survey observations (`SDSS17`, $+18.8\%$ log-loss reduction, `Gemini 3.8 Flash`).

II: Statistical and Structural Features (Φ_{feat} on Generic Schemas, 34 Datasets). On the remaining 34 datasets with anonymized headers (`f_01`–`f_N`) or generic business counters (Category II in Table 6), the agents cannot infer domain formulas from column names. Instead, the agents use Φ_{feat} to handle inputs that tabular transformers model poorly, achieving multi-fold test error reductions of $+2.31\%$ (`Opus 5`, $34/51$ datasets) and $+2.99\%$ (`Gemini 3.8 Flash`, $29/51$ datasets), and $+3.87\%$ under *Combined Best*. Three kinds of features dominate Category II:

- *Bipartite Graph Degree and Co-occurrence Features:* When a table consists of high-cardinality categorical entity IDs (such as employee roles, departments, managers, and resource IDs on `Amazon_employee_access`), standard categorical encodings assign arbitrary integer indices that hide which entities co-occur. Both agents compute label-free features on the combined train-and-test entity graph (entity frequencies, pairwise co-occurrences, and bipartite node degrees), reducing official test error $1 - \mathrm{AUROC}$ (where AUROC is the area under the receiver operating characteristic curve, and lower error is better) from 0.1416 to 0.1039 (`Opus 5`) and 0.1063 (`Gemini 3.8 Flash`) ($+26.6\%$ and $+25.0\%$ relative error reduction).
- *Frequency Encodings and Missingness Signals:* On `kddcup09_appetency`, whose 212 anonymized columns include 38 high-cardinality string columns and many columns that are mostly missing, `Opus 5` encodes each string column by its level count and a cross-fitted target mean, ranks the engineered columns by in-fold univariate AUROC with missing values treated as their own extreme (so that missingness counts as signal), and keeps only the top-ranked columns.
- *Pruning Redundant Columns and SVD on Wide Tables:* On wide cheminformatics and bioassay matrices (`Bioresponse` and `hiva_agnostic`) where over a thousand sparse molecular or bioassay columns dilute column-wise attention, the agents prune duplicate and near-collinear columns and append low-rank truncated SVD projections that summarize the highest-variance directions in a few columns.

Context Selection ($\mathcal{S}_{\text{ctx}}$). `Opus 5` and `Gemini 3.8 Flash` modify the context-selection stage $\mathcal{S}_{\text{ctx}}$ on 25.5% ($13/51$) and 23.5% ($12/51$) of datasets, respectively. When training tables exceed `TabFM`'s 16,384-row pretraining length or exhibit severe class imbalance, uniform subsampling drops rare positive instances. The agents avoid this by returning several context index sets $\{I_1, \ldots, I_K\}$: one drawn from the natural class distribution and others that oversample the minority class or stratify by cluster.

Table 7: **Final pipelines found for `TabFM` also help other tabular foundation models.** TabArena Elo of each model with its default configuration and within the final per-dataset pipelines that `TabFM-Auto` found for `TabFM` with each coding agent and LLM. Each model is rated in a separate fit per configuration, and *Gain* is measured against the default model rated in the same fit. *Default* is the mean of the three default ratings. Bold marks the largest gain for each model.

Model	Pipelines	Overall (51)		Classification (38)		Regression (13)	
		Elo	Gain	Elo	Gain	Elo	Gain
`TabICLv2`	Default	1593.0		1587.1		1726.6	
	Codex, Opus 5	1732.8	+139.4	1685.1	+97.6	2088.9	**+361.5**
	Claude Code, Opus 5	1736.7	**+143.3**	1704.4	**+116.8**	2006.5	+279.5
	Antigravity, Gemini 3.8 Flash	1700.2	+108.0	1678.2	+92.1	1938.5	+213.1
`TabPFN-3`	Default	1666.0		1643.8		1869.3	
	Codex, Opus 5	1755.5	+89.7	1705.1	+61.8	2112.2	+242.6
	Claude Code, Opus 5	1797.3	**+130.8**	1751.1	**+106.5**	2189.5	**+318.5**
	Antigravity, Gemini 3.8 Flash	1777.0	+111.2	1741.8	+98.3	2078.5	+211.3
`EXAONE-Tabular`	Default	1773.6		1770.0		1971.6	
	Codex, Opus 5	1857.5	+82.3	1839.6	**+68.4**	2088.0	+118.6
	Claude Code, Opus 5	1862.0	**+88.7**	1817.2	+46.4	2200.1	**+225.8**
	Antigravity, Gemini 3.8 Flash	1841.2	+68.8	1814.5	+46.5	2083.8	+112.7

III: Output Calibration (Ψ_{post}) and Constructor Settings (`TABFM_KWARGS`). Finally, `Opus 5` and `Gemini 3.8 Flash` tune input normalization and view weighting in `TABFM_KWARGS` on 86.3% and 98.0% of datasets, respectively, and adjust post-hoc output calibration in Ψ_{post} on 41.2% $(21/51)$ and 49.0% $(25/51)$. One possible reason is that the synthetic prior used to pretrain `TabFM` need not match the class balance and confidence levels of real datasets. In Ψ_{post}, the agents shift predicted probabilities toward the empirical training class prior $\hat{\pi}_{\text{train}}$ in log-odds space and apply temperature scaling. Category III in Table 6 contains the 5 runs (all in `Gemini 3.8 Flash`, as `Opus 5` modifies the feature table on all 51 datasets) where the agent does not change the feature matrix ($\Phi_{\text{clean}} = \Phi_{\text{feat}} = \text{Id}$), averaging $+2.16\%$ test error reduction across the 5 datasets: 4 of the 5 adjust context sampling in $\mathcal{S}_{\text{ctx}}$ and/or log-odds prior calibration in Ψ_{post} (such as `E-CommereShippingData` at $+6.94\%$ and `website_phishing` at $+3.82\%$), whereas the single run that tuned only constructor hyperparameters scored -0.30% on test.

B.3 Transferring Final Pipelines to Other Tabular Foundation Models

For each dataset, we take the final pipeline P^* that `Codex` with `Opus 5`, `Claude Code` with `Opus 5`, or `Antigravity` with `Gemini 3.8 Flash` found for `TabFM`, and run its four functions unchanged around another frozen tabular foundation model. Each ensemble member keeps the number of context rows that `TabFM` used with that pipeline. We do not carry over the other `TABFM_KWARGS` settings, because the other models do not support many of them, such as NNLS view weighting, feature crosses, and SVD features. We score every official test fold of all 51 datasets and compare with the official TabArena results of each model's default configuration. Each model with P^* from one configuration enters the Elo pool of Table 4 on its own and is rated in the same fit as its default version, so default ratings differ slightly from Table 4 and between configurations. Table 7 lists the ratings, Figure 11 compares the three configurations on overall Elo, and Figure 12 shows the pipelines of `Codex` with `Opus 5` and of `Antigravity` with `Gemini 3.8 Flash` by task type.

Figure 5 shows the pipelines from `Claude Code` with `Opus 5`, which give every model its largest overall gain (+88.7 to +143.3 Elo). Those from `Codex` with `Opus 5` come close on `TabICLv2` (+139.4 vs. +143.3) and give the largest gains for `TabICLv2` on regression and `EXAONE-Tabular` on classification. Those from `Antigravity` with `Gemini 3.8 Flash` give the smallest overall gains on `TabICLv2` and `EXAONE-Tabular`. With the pipelines of any of the three configurations, every model improves on both task types, though by less than `TabFM` gains from the same pipelines.

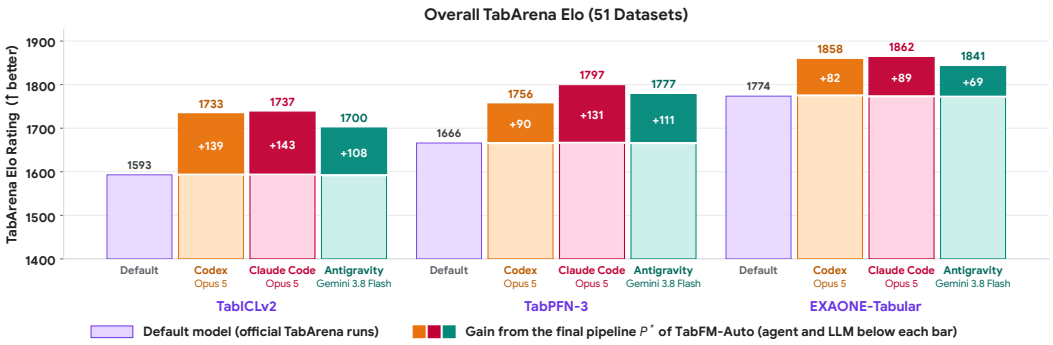

Figure 11: **The final pipelines of all three configurations help other tabular foundation models, and those from `Claude Code` with `Opus 5` help most.** Overall TabArena Elo on the 51 datasets of each model with its default configuration and within the final pipelines P^* that each `TabFM-Auto` configuration found for `TabFM`. Each model with P^* is rated in its own fit: the pale part of the bar reaches the default model rated in that fit, and the colored part is the gain. *Default* is the mean of the three default ratings.

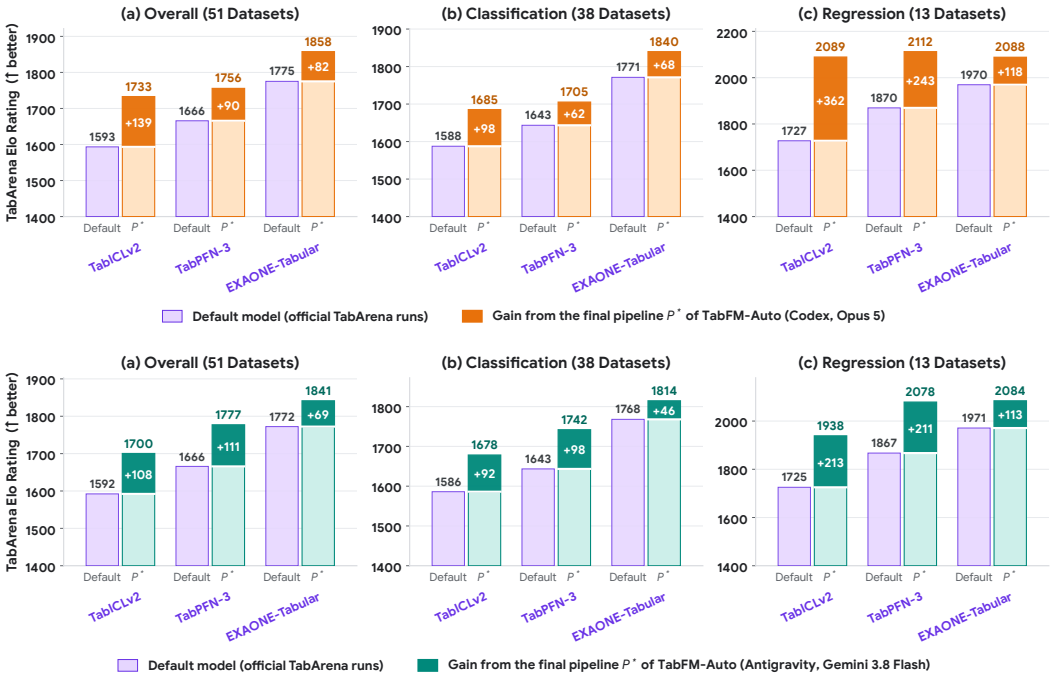

Figure 12: **The final pipelines from the other two configurations also help every model.** As in Figure 5, with the pipelines that `TabFM-Auto` found with `Codex` and `Opus 5` (top) and with `Antigravity` and `Gemini 3.8 Flash` (bottom). *Default* is the default model rated in the same fit as P^*.

B.4 Models Used by the Unconstrained Coding Agent

Figure 13 shows how often each model family appears in the code of the 51 final pipelines written by the unconstrained coding agent of Table 2. Gradient-boosted trees appear in 45 of them, mostly LightGBM (35) and CatBoost (27). No final pipeline uses XGBoost, and only 6 use a neural network. Twenty-three pipelines rely on a single model family. The other 28 blend two or more.

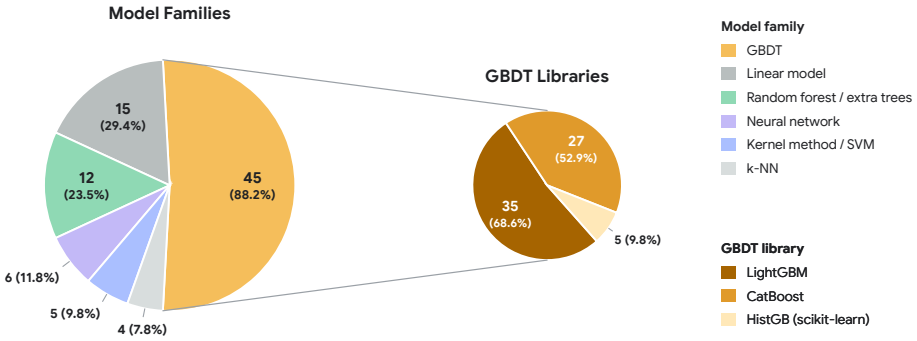

Figure 13: **Model families in the final pipelines of the unconstrained coding agent** (`Antigravity` with `Gemini 3.8 Flash`, no `TabFM`). Each wedge gives the number of final pipelines, out of 51, whose code uses that family, and in parentheses that number as a percentage of the 51 pipelines. Since a pipeline that blends several families counts once for each, the percentages sum to more than 100%. The right pie splits the GBDT wedge by library.

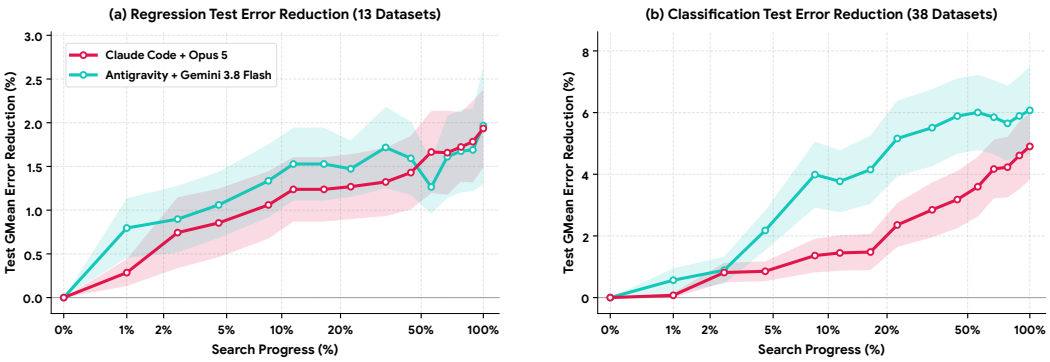

Figure 14: **Official test G-Mean error reduction across search progress (log scale).** (a) Regression suite (13 datasets) and (b) Classification suite (38 datasets).

C Search Dynamics and Generalization Analysis

We saved every candidate pipeline evaluated during the TabArena runs of `Claude Code` with `Opus 5` and `Antigravity` with `Gemini 3.8 Flash`, and rescored up to 15 checkpoints per run, taken at fixed fractions of the search budget (1,486 checkpoints in total), on the official test sets across all folds to test whether repeated 3-fold cross-validation evaluations on fold 0's training set cause validation overfitting.

Figure 14 plots the official test G-Mean error reduction relative to P_0 separately for the 13 regression datasets (Figure 14a) and the 38 classification datasets (Figure 14b) over log-scaled search progress. On both suites, official test error across all folds decreases together with the search validation error from the first evaluation to the end of the budget (100% progress). On regression, both configurations make rapid early gains in the first 10% of search as they add target transforms and primary physical ratios, then refine to 1.93% (`Opus 5`) and 1.97% (`Gemini 3.8 Flash`) lower G-Mean test RMSE than P_0 at full budget (3.11% and 3.15% lower than the published `TabFM` results in Table 9). On classification, official test error reduction climbs throughout the budget as agents add group aggregations, string decompositions, and prior calibration, reaching 4.90% (`Opus 5`) and 6.07% (`Gemini 3.8 Flash`) lower G-Mean test classification error than P_0 at 100% of the budget.

In Figure 15, we examine validation-to-test transfer within each quarter of the search budget (0%–25%, 25%–50%, 50%–75%, and 75%–100%) by plotting marginal official test error reduction (Δ%) across all folds against marginal 3-fold cross-validation gain (Δ%) on fold 0's training set during that quarter for each dataset. In most quarters, most datasets with a validation gain also improve on test (upper-



Figure 15: **Stage-wise validation-to-test generalization across the four search quarters (0%–25%, 25%–50%, 50%–75%, and 75%–100%).** Official test error reduction ($\Delta\%$) across all folds versus 3-fold cross-validation gain ($\Delta\%$) on fold 0's training set per dataset.

right quadrant), and datasets that have plateaued lie near the origin. The exception is the 50%–75% quarter of `Gemini 3.8 Flash`, where test changes split roughly evenly. In the final quarter (75%–100% of budget), validation and test gains remain positively correlated across the datasets whose validation score still improves. Keeping `TabFM` fixed and evaluating candidate edits via 3-fold cross-validation on $\mathcal{D}_{\text{train}}^{(0)}$ helps the discovered pipelines generalize across the held-out test folds.

D MLE-Bench-Tabular Analysis and Per-Competition Results

D.1 Competition-by-Competition Analysis on MLE-Bench-Tabular

Figure 16 plots the 3-fold cross-validation curves on the training split and the final official test scores across the 8 Kaggle competitions of MLE-Bench-Tabular. Table 8 summarizes the data formats, evaluation metrics, baseline scores, final test results, and main features engineered by `TabFM-Auto` (`Claude Code` with `Opus 5`) and `TabFM-Auto` (`Antigravity` with `Gemini 3.8 Flash`). Figure 7 reports overall pairwise Elo ratings computed across all graded submissions against the 12 external MLE agents with complete 8-competition submissions from the official MLE-Bench leaderboard (`PiEvolve 24h` only has public submissions on 5 of the 8 competitions and, if rated on those 5, scores 1668 Elo, still behind both `TabFM-Auto` configurations). `TabFM-Auto` (`Claude Code` with `Opus 5`) ranks #1 with 1827 Elo (86.7% win rate across all graded matches) and `TabFM-Auto` (`Antigravity` with `Gemini 3.8 Flash`) ranks #2 with 1734 Elo (80.0% win rate).

Geophysics and Multi-Sensor Waveforms (`predict-volcanic-eruptions`). The main training table in `predict-volcanic-eruptions` contains only segment IDs and time-to-eruption targets, while the sensor readings are stored in thousands of external 10-minute, 100 Hz ten-channel seismic CSV files (60,001 rows per file). The starting identity pipeline P_0 therefore fails on step 1 with zero feature columns. The coding agent then inspects the error and writes an initial summary-statistics script on step 2 (scoring a validation mean absolute error, or MAE, of 667,762 for `Gemini 3.8 Flash` and 339,254 for `Opus 5`). It then refines Φ_{feat} into a parallelized signal-processing pipeline that computes multi-band Fast Fourier Transform (FFT) spectral energy across volcanic tremor bands (0.5–20 Hz), Short-Term Average to Long-Term Average (STA/LTA) trigger ratios across sub-windows, rolling peak-to-peak envelopes, and inter-sensor cross-correlations. Passing this feature table to frozen `TabFM` lowers official test MAE to 326,950 (`Gemini 3.8 Flash`) and 97,355 (`Opus 5`), reducing error by $9.4\times$ over the strongest external agent (`PiEvolve` at 0.92M, `MLEvolve` at 1.24M) on `MLE-Bench`'s official local split and earning a Gold Medal.

Materials Science and 3D Crystal Lattices (`nomad2018-predict-transparent-conductors`). The task is to predict formation energy and bandgap energy (scored by mean column-wise root mean squared logarithmic error, or RMSLE) of $(\text{Al}_x\text{Ga}_y\text{In}_z)_2\text{O}_3$ transparent conductors. The competition provides lattice vectors $(a, b, c, \alpha, \beta, \gamma)$ in the main table alongside external `geometry.xyz` files containing 3D Cartesian coordinates of the atoms in each unit cell. `TabFM-Auto` parses the 3D crystal files to compute the parallelepiped unit-cell volume $V = abc\sqrt{1 + 2\cos\alpha\cos\beta\cos\gamma - \cos^2\alpha - \cos^2\beta - \cos^2\gamma}$, atomic packing density, reciprocal

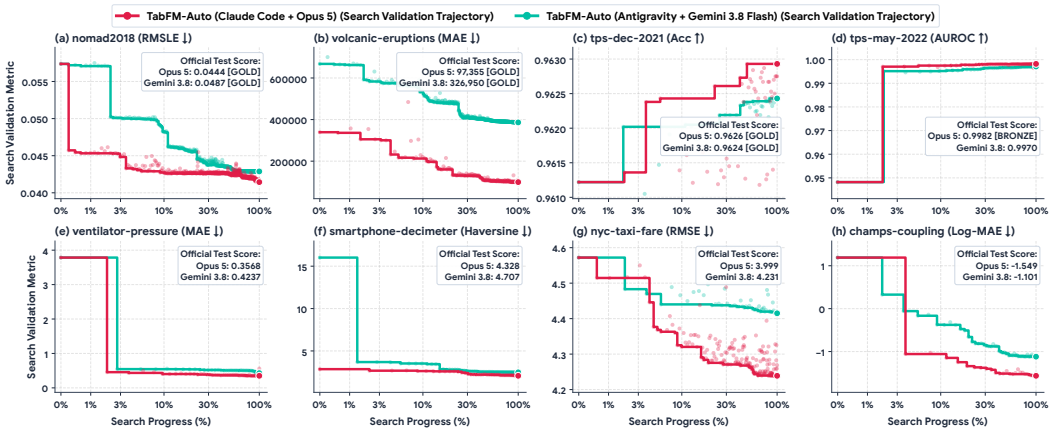

Figure 16: **Search progression on MLE-Bench-Tabular.** Best-so-far 3-fold cross-validation curves on the training split and final official test scores across the 8 Kaggle competitions (on `volcanic-eruptions` and `smartphone-decimeter`, where P_0 fails with zero features, 0% plots each agent's step-2 initial multi-file script).

lattice constants, and composition-weighted Pauling electronegativity and ionic radius variance across the Al/Ga/In cations. These physical features reduce official test RMSLE from 0.05737 (`TabFM`) to 0.04869 (`Gemini 3.8 Flash`) and 0.04440 (`Opus 5`), earning a Kaggle Gold Medal and ranking #1 among all evaluated agents.

Quantum Chemistry and 3D Molecular Coupling (`champs-scalar-coupling`). Predicting nuclear magnetic resonance (NMR) spin-spin scalar coupling constants (J-coupling, evaluated by mean Log-MAE across 8 coupling types such as `1JHC` and `3JHH`) requires joining the bond-pair table with 3D molecular coordinates in `structures.csv`. Without 3D geometry, `TabFM` scores $+1.19025$ Log-MAE. `TabFM-Auto` writes a geometry pipeline in Φ_{feat} that computes 3D inter-atomic distances r_{ij} and inverse power laws ($r_{ij}^{-1}, r_{ij}^{-2}, r_{ij}^{-3}$), covalent bond-path angles, and four-atom dihedral torsion angles ϕ from the Karplus relation ($\cos\phi$ and $\cos^2\phi$) for vicinal `3J` couplings, while routing each coupling type through stratified context windows in $\mathcal{S}_{\text{ctx}}$. This lowers official test Log-MAE to -1.10133 (`Gemini 3.8 Flash`) and -1.54947 (`Opus 5`).

Respiratory Control Dynamics (`ventilator-pressure-prediction`). This competition provides breath-by-breath time series (80 time steps per breath) of inspiratory solenoid valve opening percentages $u_{\text{in}}(t) \in [0, 100]$, expiratory valve states $u_{\text{out}}(t)$, and lung constants (R resistance and C compliance) to predict airway pressure. Treating time steps as independent rows in `TabFM` gives 3.78450 MAE because it ignores the accumulated air in the lung. Inspired by the single-compartment respiratory mechanics relation $P(t) = P_0 + Rq(t) + \frac{1}{C}\int_0^t q(\tau)\,d\tau$, `TabFM-Auto` uses $u_{\text{in}}(t)$ as a flow proxy to compute within-breath cumulative integrals $\int_0^t u_{\text{in}}(\tau)\,d\tau$, first and second finite differences ($\Delta u_{\text{in}}, \Delta^2 u_{\text{in}}$), decayed flow histories, and proxy interaction terms ($R \times u_{\text{in}}$ and $\frac{1}{C}\int_0^t u_{\text{in}}(\tau)\,d\tau$). These state features reduce official test MAE to 0.42369 (`Gemini 3.8 Flash`) and 0.35684 (`Opus 5`, a 90.6% MAE reduction).

GNSS Satellite Navigation (`smartphone-decimeter-2022`). Predicting phone positions (scored by the mean of the 50th and 95th percentile horizontal Haversine distance errors in meters across trips) from raw Android Global Navigation Satellite System (GNSS) receiver logs requires joining drive traces with multi-gigabyte satellite pseudorange and Inertial Measurement Unit (IMU) tables (with ground-truth NMEA logs excluded). After the zero-feature identity pipeline P_0 fails on step 1, the coding agent extracts an initial weighted least squares (WLS) baseline from the receiver logs on step 2 (16.010 m validation error for `Gemini 3.8 Flash` and 2.850 m for `Opus 5`). It then computes elevation-weighted carrier-to-noise density (C/N_0) averages, pseudorange residual dispersion, and forward-backward Kalman velocity-smoothed positions across consecutive epochs,

Table 8: **End-to-end performance on 8 Kaggle MLE-Bench competitions.** Best score per task is **bolded**, second best is underlined.

Competition	Domain & Data Format	Metric	Identity TabFM	AGY + Gemini 3.8 Flash	CC + Opus 5	Result	Key Features Added in `engineer()`
`volcanic-eruptions`	Geophysics (Waveforms)	MAE ↓	Fails (IDs)†	326,950	**97,355**	GOLD	Multi-sensor waveform FFT bands, STA/LTA ratios, quantiles (9.4× < `PiEvolve`)
`nomad2018-conductors`	Materials (3D Lattice)	RMSLE ↓	0.05737	0.04869	**0.04440**	GOLD	Reciprocal lattice unit-cell volume, cation electronegativity & radius dispersion
`tps-dec-2021`	Cartography & Hydrology	Acc ↑	0.96122	0.96237	**0.96264**	GOLD	Euclidean hydrology distance $\sqrt{H^2+V^2}$, aspect compass decomposition
`tps-may-2022`	Manufacturing Telemetry	1 − AUROC ↓	0.05179	0.00304	**0.00179**	BRONZE	Decomposing 10-char string `f_27` into positional ordinal & unique-char counts
`champs-coupling`	Quantum Chem (3D NMR)	Log-MAE ↓	+1.19025	−1.10133	**−1.54947**	>MEDIAN	3D inter-atomic Euclidean distance r_{ij}^{-3}, Karplus dihedral $\cos^2\phi$ angles
`ventilator-pressure`	Respiratory Control	MAE ↓	3.78450	0.42369	**0.35684**	−90.6%	Respiratory mechanics proxy integrals ($\int u_{\text{in}}\,dt$, Δu_{in}, $R \times u_{\text{in}}$)
`smartphone-decimeter`	GNSS Navigation	Haversine ↓	Fails (IDs)†	4.7068	**4.3284**	>MEDIAN	Multi-table GNSS pseudorange WLS + IMU Kalman smoothing (18.9% < `Famou-Agent 2.0`)
`nyc-taxi-fare`	Spatial Econometrics	RMSE ↓	4.57207	4.23091	**3.99861**	−12.5%	Geodesic Haversine distance, Manhattan rotated grid, JFK/LGA/EWR polygons

†Raw tabular identity (P_0) fails on `volcanic-eruptions` and `smartphone-decimeter` because the main table contains only recording IDs; on step 2 the coding agent inspects the error and writes an initial multi-file script itself (scoring 667,762 and 339,254 validation MAE on `volcanic-eruptions`, and 16.010 m and 2.850 m validation Haversine error on `smartphone-decimeter`, for `Gemini 3.8 Flash` and `Opus 5`, respectively) before refining it.

reducing official test Haversine error to 4.7068 m (`Gemini 3.8 Flash`) and 4.3284 m (`Opus 5`, ranking #1 overall and 18.9% below `Famou-Agent 2.0` at 5.334 m).

Cartography, Manufacturing Telemetry, and Spatial Econometrics (`tps-dec-2021`, `tps-may-2022`, and `nyc-taxi-fare`). On the remaining three large-scale tabular competitions, `TabFM-Auto` extracts geometric and structural features from the schema. All scores below are for `Opus 5`. On `tabular-playground-series-dec-2021`, computing Euclidean distance to hydrology $\sqrt{d_{\text{horiz}}^2 + d_{\text{vert}}^2}$, relative elevation above hydrology, and compass aspect $(\sin\theta, \cos\theta)$ raises official test accuracy to 0.96264 (Gold Medal). On `tabular-playground-series-may-2022`, splitting the 10-character manufacturing string `f_27` into 10 positional ASCII ordinal columns, unique character counts, and continuous interaction terms reduces test $1 - \text{AUROC}$ error from 0.05179 to 0.00179 (Bronze Medal). On `new-york-city-taxi-fare-prediction`, computing Haversine and 29°-rotated Manhattan street-grid distances along with bounding-box flags for JFK, LaGuardia, and Newark airport flat-rate zones reduces test RMSE by 12.5% from 4.57207 to 3.99861.

E Per-Dataset Official Test Scores on TabArena

Table 9 reports the per-dataset official test scores across all 51 TabArena datasets, grouped into the 13 regression datasets evaluated by RMSE (Part I), the 30 binary classification datasets evaluated by $1 - \text{AUROC}$, for which lower is better (Part II), and the 8 multiclass classification datasets evaluated by multiclass log-loss (Part III).

On Part I (Regression), `TabFM-Auto` improves official test RMSE over `TabFM` on all 13 datasets, achieving a suite G-Mean RMSE reduction of +3.11% (`Opus 5`) and +3.15% (`Gemini 3.8 Flash`) compared to +1.35% for `TabFM+`. The largest relative RMSE reductions occur on physical and engineering tasks where column names indicate domain formulas, including `airfoil_self_noise` (+17.3%, `Gemini 3.8 Flash`), `physiochemical_protein` (+3.9%, `Opus 5`), `concrete_strength` (+3.8%, `Opus 5`), and `miami_housing` (+3.1%, `Opus 5`).

On Part II (Binary Classification), `TabFM-Auto` achieves a suite G-Mean reduction in test classification error ($1 - \text{AUROC}$) of +3.51% (`Opus 5`) and +5.17% (`Gemini 3.8 Flash`) vs. +0.81% for `TabFM+`. The largest reductions in $1 - \text{AUROC}$ occur on `Amazon_employee_access` (0.1416 → 0.1039, `Opus 5`), `churn` (0.0681 → 0.0491, `Gemini 3.8 Flash`), `qsar-biodeg` (0.0580 → 0.0413, `Gemini 3.8 Flash`), and `Marketing_Campaign` (0.0732 → 0.0581, `Gemini 3.8 Flash`). Error increases are small (at most +0.0034 in absolute $1 - \text{AUROC}$). The largest relative increases occur on nearly saturated datasets where `TabFM`

Table 9: **Per-Dataset Official Test Performance Across All 51 TabArena Benchmarks.** For each dataset, we report the number of official evaluation folds, `TabFM`, and the final submitted pipelines discovered by `TabFM-Auto` (`Claude Code` with `Opus 5`) and `TabFM-Auto` (`Antigravity` with `Gemini 3.8 Flash`) across the Regression (RMSE ↓), Binary Classification ($1 - \text{AUROC}$ ↓), and Multiclass Classification (Log-Loss ↓) suites. Parentheses report the relative test metric improvement ($\Delta\%$ RMSE reduction, $1 - \text{AUROC}$ reduction, or Log-Loss reduction) vs. TabFM. Best score per row is **bolded**.

Dataset	Folds	TabFM	TabFM-Auto (Opus 5)	TabFM-Auto (Gemini 3.8 Flash)
Part I: Regression Suite (13 Datasets - Metric: RMSE ↓)				
airfoil_self_noise	30	1.0734	0.91650 (+14.6%)	**0.88780** (+17.3%)
Used-Fiat-500	30	703.27	**693.49** (+1.4%)	696.52 (+1.0%)
concrete_strength	30	3.9666	**3.8169** (+3.8%)	3.8309 (+3.4%)
diamonds	9	496.42	**484.52** (+2.4%)	484.54 (+2.4%)
Food_Delivery_Time	9	7.3328	7.1736 (+2.2%)	**7.1724** (+2.2%)
healthcare_insurance	30	4,417.6	**4,344.1** (+1.7%)	4,392.3 (+0.6%)
houses	9	0.18155	0.17706 (+2.5%)	**0.17696** (+2.5%)
miami_housing	9	74,069.5	**71,802.2** (+3.1%)	72,037.1 (+2.7%)
physiochemical_protein	9	2.8875	**2.7751** (+3.9%)	2.7838 (+3.6%)
QSAR-TID-11	9	0.73934	0.72710 (+1.7%)	**0.72372** (+2.1%)
QSAR_fish_toxicity	30	0.85345	**0.84829** (+0.6%)	0.85218 (+0.1%)
superconductivity	9	8.9437	**8.7946** (+1.7%)	8.8023 (+1.6%)
wine_quality	9	0.58651	**0.58551** (+0.2%)	0.58622 (+0.0%)
Suite G-Mean RMSE Reduction vs. TabFM	—	—	**+3.11%**	**+3.15%**
Part II: Binary Classification Suite (30 Datasets - Metric: $1 - \text{AUROC}$ ↓)				
Amazon_employee_access	9	0.1416	**0.1039** (+26.6%)	0.1063 (+25.0%)
APSFailure	9	0.0056	**0.0054** (+3.0%)	**0.0054** (+3.0%)
bank-marketing	9	0.2310	0.2314 (-0.2%)	**0.2303** (+0.3%)
Bank_Customer_Churn	9	0.1229	0.1228 (+0.1%)	**0.1225** (+0.3%)
Bioresponse	9	0.1192	0.1171 (+1.8%)	**0.1168** (+2.0%)
blood-transfusion-service-center	30	0.2441	0.2431 (+0.4%)	**0.2398** (+1.8%)
churn	9	0.0681	0.0610 (+10.5%)	**0.0491** (+27.9%)
coil2000_insurance	9	0.2276	**0.2127** (+6.5%)	0.2225 (+2.2%)
credit-g	30	0.1944	**0.1940** (+0.2%)	0.1978 (-1.8%)
credit_card_default	9	**0.2068**	0.2073 (-0.2%)	0.2073 (-0.2%)
customer_satisfaction_in_airline	9	**0.0038**	0.0038 (-0.9%)	0.0038 (-1.6%)
diabetes	30	0.1580	**0.1471** (+6.9%)	0.1550 (+1.9%)
Diabetes130US	9	0.3292	**0.3193** (+3.0%)	0.3207 (+2.6%)
E-CommereShippingData	9	0.2592	0.2508 (+3.2%)	**0.2412** (+6.9%)
Fitness_Club	30	0.1789	0.1787 (+0.1%)	**0.1785** (+0.2%)
GiveMeSomeCredit	9	0.1321	0.1305 (+1.2%)	**0.1303** (+1.3%)
hazelnut-spread-contaminant	30	0.0023	**0.0021** (+6.7%)	0.0021 (+6.4%)
heloc	9	0.1976	0.1973 (+0.2%)	**0.1971** (+0.3%)
HR_Analytics_Job_Change	9	0.1935	**0.1933** (+0.1%)	0.1940 (-0.3%)
in_vehicle_coupon	9	0.1513	0.1405 (+7.2%)	**0.1402** (+7.3%)
Is-this-a-good-customer	30	0.2466	**0.2432** (+1.4%)	0.2499 (-1.3%)
jm1	9	0.2109	0.2114 (-0.3%)	**0.2094** (+0.7%)
kddcup09_appetency	9	0.1587	**0.1505** (+5.2%)	0.1515 (+4.5%)
Marketing_Campaign	30	0.0732	0.0616 (+15.8%)	**0.0581** (+20.7%)
NATICUSdroid	9	**0.0112**	0.0115 (-2.6%)	0.0122 (-9.3%)
online_shoppers_intention	9	**0.0601**	0.0602 (-0.3%)	0.0605 (-0.7%)
polish_bankruptcy	9	**0.0049**	0.0052 (-6.5%)	0.0054 (-11.5%)
qsar-biodeg	30	0.0580	0.0582 (-0.3%)	**0.0413** (+28.7%)
seismic-bumps	9	0.2047	**0.1987** (+2.9%)	0.2022 (+1.2%)
taiwanese_bankruptcy	9	0.0494	0.0459 (+7.1%)	**0.0395** (+20.0%)
Suite G-Mean ($1 - \text{AUROC}$) Reduction vs. TabFM	—	—	**+3.51%**	**+5.17%**
Part III: Multiclass Classification Suite (8 Datasets - Metric: Log-Loss ↓)				
anneal	30	0.0125	**0.0103** (+17.3%)	0.0114 (+8.9%)
hiva_agnostic	9	0.1787	0.1779 (+0.4%)	**0.1742** (+2.5%)
maternal_health_risk	30	0.3706	0.3648 (+1.6%)	**0.3610** (+2.6%)
MIC	30	0.4282	**0.4178** (+2.4%)	0.4199 (+1.9%)
SDSS17	9	0.0693	0.0564 (+18.6%)	**0.0563** (+18.8%)
splice	9	0.0960	**0.0747** (+22.2%)	0.0757 (+21.1%)
students_dropout_and_academic_success	9	**0.5068**	0.5111 (-0.8%)	0.5132 (-1.2%)
website_phishing	30	0.2104	0.2067 (+1.8%)	**0.2024** (+3.8%)
Suite G-Mean Log-Loss Reduction vs. TabFM	—	—	**+8.39%**	**+7.64%**
Overall G-Mean Error Reduction (All 51 Datasets)	—	**0.00%**	**+4.19%**	**+5.05%**

already attains $1 - \text{AUROC}$ around or below 0.01 (`customer_satisfaction_in_airline` at 0.0038, `polish_bankruptcy` at 0.0049, `NATICUSdroid` at 0.0112), and the largest absolute increases occur on small, noisy credit tables (`credit-g` with $N = 1{,}000$, and `Is-this-a-good-customer`), where validation gains on fold 0 do not fully transfer across small evaluation folds. All other increases are at most $+0.0005$.

On Part III (Multiclass Classification), TabFM-Auto achieves its largest suite-level G-Mean improvement, reducing official test log-loss by $+8.39\%$ (Opus 5) and $+7.64\%$ (Gemini 3.8 Flash) compared to $+1.50\%$ for TabFM+, improving 7 out of 8 datasets. Combining domain feature engineering in Φ_{feat} with multiclass temperature scaling and log-odds prior alignment in Ψ_{post} produces the largest test log-loss reductions on splice ($+22.2\%$, Opus 5), SDSS17 ($+18.8\%$, Gemini 3.8 Flash), anneal ($+17.3\%$, Opus 5), and website_phishing ($+3.8\%$, Gemini 3.8 Flash). On all 51 datasets, TabFM-Auto reduces overall G-Mean test error by $+4.19\%$ (Opus 5) and $+5.05\%$ (Gemini 3.8 Flash), roughly four to five times the $+1.06\%$ error reduction of feature engineering that ignores column meanings (cross and SVD features) and test-time ensembling in TabFM+.